Visual Informatics 5 (2021) 54–69

Contents lists available at ScienceDirect

Visual Informatics

journal homepage: www.elsevier.com/locate/visinf

Algorithms for rapid digitalization of prescriptions

Mehul Gupta *, Kabir Soeny

1mg Technologies Private Limited, Gurugram, India

ARTICLE INFO

Article history:
Received 14 April 2021
Received in revised form 8 June 2021
Accepted 12 July 2021
Available online 17 July 2021

Keywords:
Artificial intelligence
Image processing
Healthcare technology
Pattern intelligence

ABSTRACT

Prescription data are invaluable for healthcare research and intelligence, yet, extraction of these data is challenging as this information is intertwined in the unstructured and non-grammatical text in prescription images. Moreover, text extraction from images in itself is hard, particularly for handwritten text. While piecemeal solutions exist, they are either limited to a small set of entities of interest or have very low accuracy and are not scalable. In this paper, we present two algorithms: the *C-Cube algorithm* for digitization of computer-printed prescriptions and the *3-Step Filtering* algorithm for handwritten prescriptions. While a brute-force approach would match every word that is received from an optical character reader (OCR) with all possible entries in the database, this approach is inefficient and imprecise. The premise of our algorithms is an application of pattern intelligence to select a much smaller set of words (from the words returned by the OCR) as potential entities of interest. We rigorously tested the two algorithms on a corpus of more than 10,000 prescriptions' images, taking the brute-force technique as the baseline methodology. Regarding latencies, we found that the C-Cube and the 3-Step Filtering algorithms were 588 and 231 times faster than the brute-force approach. In terms of accuracies, we found that the F-score of the C-cube algorithm was 90% higher than the F-score of the brute-force approach whereas the F-score for the 3-Step filtering algorithm was found to be 8,600% higher. The algorithms are decidedly faster and more accurate than the brute-force approach. These attributes make them suitable for implementation in real-time environments as well as for use in batch-mode for various applications. We expect the algorithms to play a significant role in the digitalization of healthcare information and briefly discuss a few applications.

1. Introduction

With the phenomenon of big data permeating the healthcare industry as well, there are unprecedented opportunities for the development of advanced solutions for some of the most complex and impactful problems in healthcare. For example, it is estimated that the global patient healthcare data generated in 2020 were of the size of about 25 Exabytes (1 Exabyte = 10^{18} bytes), Chrimes et al. (2016). With the generation of such a humongous amount of information, which is often unstructured and paper-based (such as prescriptions, diagnostic reports, discharge summaries), transforming these into digital formats through efficient and scalable technologies can catalyze the development of AI-based products that can be of benefit to not just the individual patients but the broader healthcare research community as well. Furthermore, studies suggest that patients often misunderstand the information contained in prescriptions or lab reports due to a lack of medical knowledge and the inability to correctly read them, (Davis et al., 2008; Medicine et al., 2000). Effective digitalization of such health records could, therefore, spur the development of, for example, smartphone-based apps which could assist the patient community with such problems.

A prescription is a doctor's order which stipulates the administration of drugs in the specified amount, duration and frequency, and contains details of the patient such as name, age and gender, and also the details of the doctor who writes the prescription. A prescription ensures that the information about the prescribed drugs is accurately passed on to a pharmacist who then ensures that the drugs are dispensed without any errors. Such prescriptions can either be completely computer-printed (henceforth referred to as a *printed prescription*) or handwritten by the doctor on a printed letterhead (henceforth referred to as *a handwritten prescription*). It is a common observation that a doctor's handwriting is difficult to understand, Lyons et al. (1998). While this implies that a printed prescription is easier to read for a human eye as compared to a handwritten prescription, we will discuss later in the paper that this difference applies to the reading by the machine as well.

Our organization, 1mg Technologies, 1mg Technologies (2021), is a digital healthcare platform, which offers services such as the electronic ordering of medicines, healthcare products and diagnostic tests, and tele-consultations. As part of the business

* Corresponding author.

https://doi.org/10.1016/j.visinf.2021.07.002

processes, we digitize prescriptions' images that are uploaded to our platform by the users, mainly extracting the names of the medicines contained in that image. Given that we receive more than ten million orders annually, we need to invest thousands of man-hours in the task of searching for the relevant information in every prescription. This information is then securely stored for every prescription against the corresponding user identification key. We have been motivated to develop solutions that make this process faster by having a machine predict the names of the medicines in the prescription, as an assistive technology, without compromising on the accuracy standards.

In our review of the existing literature on this problem, we observed sparsity in the works that cover both prescriptions' types, as well as, work for a large universe of medicines' names. Most of the past works focus on OCR-based solutions (image taken as an input to, say, a deep learning model) especially for handwritten prescriptions where the text is hard to interpret. For example, Fajardo et al. (2019) created a special dataset where each image has just one medicine name and unnecessary information present in the prescription images was eliminated. A set of twelve unique medicines was considered and a hybrid neural network consisting of a CNN with RNN, that is, CRNN was trained. Even on a small set of only twelve medicines, the validation accuracy was only 35% which highlights the challenges associated with machine recognition of a doctor's handwriting. Furthermore, annotated data were used for training of the model, which are tedious to prepare.

There are a few works on this problem that emanated from the Informatics for Integrating Biology and the Bedside's (i2b2) 3rd workshop on Natural Language Processing challenges for clinical records, Uzuner et al. (2010). As part of this workshop, a dataset comprising of pre-annotated discharge summaries was used and the problem statement was the extraction of entities such as medicine names, dosage, frequencies. Multiple solutions were developed for the given dataset and the problem statement. For example, (Tao et al., 2017) approached the problem by using conditional random fields (CRFs) with Part-of-Speech (POS) tagging as the major feature. Post-processing included the use of exact matching with line sequences from the drug category of Wikipedia and Google search API to boost the recall metric. On the other hand, (Li et al., 2010) used CRFs as well as developed an AdaBoost classification model for associating a medication name with its corresponding fields (such as dosage, frequency, reason) and then using a support vector machine (SVM) model for distinguishing between different writing styles. Patrick and Li (2009) developed a cascade approach for extraction of medication events based on the combination of a machine learning approach and a rule-based approach. Two machine learners were used, namely a CRF for medication and other meta-information labelling and SVM for relationship extraction. A number of rule-based systems were also developed. Rules such as dividing the entire text into different regions and then rule-based tagging, regex patterns, exact token matching, using an external vocabulary like UMLS Metathesaurus were used, Grouin et al. (2009), Solt and Tikk (2009), Yang (2009), Doan et al. (2009), Hamon and Grabar (2009), Spasić et al. (2010) and Bodenreider (2004).

However, the methods described above do not help our problem statement because of a variety of reasons. Firstly, they were developed on a small set of discharge summaries which are structurally and content-wise quite different from a typical prescription. Secondly, as these were pre-annotated manually, the effect of errors in the OCR output was not considered. Thirdly, the latencies of these approaches were not evaluated. Finally, the text present in a discharge summary has some grammatical structure which is generally absent in a prescription.

The work presented in this paper is a step in the direction of filling the above gaps. Firstly, we present dedicated algorithms for both prescriptions' types: the *C-Cube algorithm* for digitization of printed prescriptions and the *3-Step Filtering* algorithm for handwritten prescriptions. Secondly, our algorithms are not limited to a small set of medicines' names but can work for a database of any size. We have evaluated our algorithms using our own database of stock keeping units (SKUs) which consists of more than 95,000 medicines' names. Thirdly, we present clear metrics about the accuracies of the algorithms and establish their effectiveness. Fourthly, we demonstrate that our algorithms are decidedly faster and are suitable for use in a real-time production environment as well as for use in the batch-mode. Finally, our algorithms do not require any annotated datasets for training and can be used, therefore, on the plug-and-play basis.

Our algorithms are built over the output that is received from a state-of-the-art OCR. While a brute-force approach would match every word that is received from the OCR with all the possible medicines' names, this approach is inefficient and imprecise. The premise of our algorithms is the application of pattern intelligence to select a much smaller set of words (from the words returned by the OCR) as potential entities of interest. This approach results in very low latencies and substantially improved F-scores. We expect the algorithms to play a significant role in the digitalization of healthcare information.

The paper is organized as follows: In Section 2, we describe the three approaches that we discuss in this paper — the brute-force, the C-cube and the 3-step filtering algorithms, along with their implementation details. Section 3 presents the details of our experiments, along with the results pertaining to latencies and accuracies for the three approaches. Finally, in Section 4, we discuss a few extensions, limitations and improvement opportunities for our algorithms. We also briefly discuss a few applications of the algorithms.

2. Methods and motivation

2.1. Google Vision API

Google Cloud's Vision API offers powerful pre-trained machine learning models through REST APIs, Google (2021a). It assigns labels to images and is capable of detecting objects and reading printed and handwritten text.

The prescriptions' digitization algorithms that we have developed consume the output that is returned from the Vision API. Other OCR could also be used, provided they return not just the read words, but also their co-ordinates (relative position in the image).

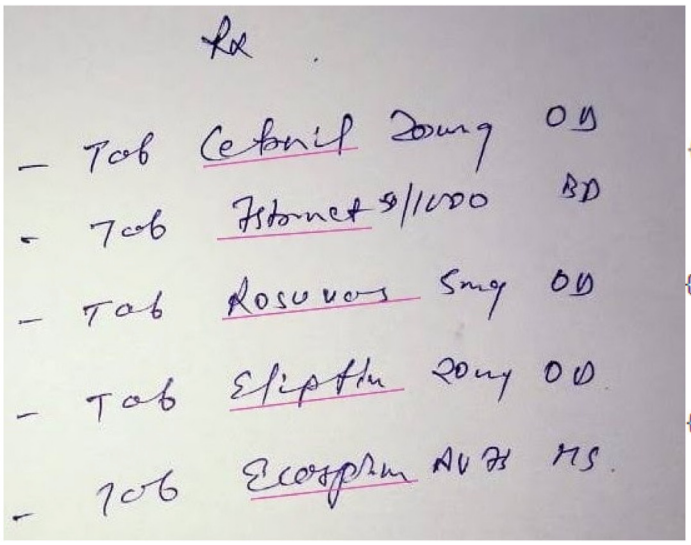
We briefly discuss one of the components of the output from the API which is pertinent to our work. *FullTextAnnotation*, is a structured hierarchical response for the UTF-8 text extracted from the image. The hierarchy is organized in the following order: Pages > Blocks > Paragraphs > Words > Symbols.

In this paper, we would only be using the 'Words' that are contained in the response and their associated co-ordinates. We shall, therefore, discuss them briefly here.

Word is the smallest unit of text, represented as an array of Symbols. A word provides the following information: four (x, y) co-ordinate pairs starting from top left of the word, the associated confidence scores and symbols.

Fig. 1 presents some of the output from FullTextAnnotation for part of a prescription's image uploaded on our platform.

The co-ordinate system according to which co-ordinates are provided assumes (0, 0) at the top left corner of the image irrespective of its orientation. The four (x, y) co-ordinate pairs in every entity start from the top left corner of the entity (and not the image) and the sequence for the rest of the 3 coordinate pairs are given in a clock-wise order. Hence, the same image with




```
{'description': 'Istornet',
 'boundingPoly': {'vertices': [{'x': 575, 'y': 540},
   {'x': 665, 'y': 533},
   {'x': 668, 'y': 580},
   {'x': 578, 'y': 587}]}},
{'description': 'Kosovom',
 'boundingPoly': {'vertices': [{'x': 563, 'y': 607},
   {'x': 679, 'y': 600},
   {'x': 682, 'y': 647},
   {'x': 566, 'y': 654}]}},
{'description': 'Eliptin',
 'boundingPoly': {'vertices': [{'x': 576, 'y': 679},
   {'x': 687, 'y': 673},
   {'x': 689, 'y': 716},
   {'x': 578, 'y': 722}]}},
{'description': 'Ecospem',
 'boundingPoly': {'vertices': [{'x': 589, 'y': 744},
   {'x': 701, 'y': 735},
   {'x': 705, 'y': 793},
   {'x': 593, 'y': 802}]}},
```


Fig. 1. Partial image of a prescription uploaded at our platform and the output returned from the Google Vision API. Notice that every word is accompanied by four pairs of (x, y) co-ordinates.

different orientations (rotated at different angles) will provide different co-ordinates for the same entities.

These words and the associated co-ordinates that are received from the OCR are essential ingredients of our methodology that we describe later in the paper. While only the words that contain the names of the medicine are of interest, as they are mixed with other irrelevant words extracted from the prescription, it is non-trivial to isolate them. While a brute-force technique could match every OCR word with every medicine name available, this approach is inefficient and vulnerable to false positives, especially when fuzzy matching techniques are used.

This is where the associated co-ordinates of the words can be leveraged. The co-ordinates of a word do not just indicate the relative position of that word in the prescription, but the co-ordinates can be used to compute the area bound by the word, which is an important signal about the potential nature of that word. For example, for the same number of letters, a word that is printed in the prescription will, in general, occupy a lesser area than a word which is handwritten. We discuss these ideas later when we describe the algorithms that we developed to separate the potential medicines' names, referred to as *candidate words* from the words that are received from the OCR.

We firstly present the most basic approach for digitization, the brute-force approach.

2.2. The brute-force approach

The problem statement is that given an image of a prescription, predict the medicines' names contained in that prescription. The prescription could be either completely printed or could have the names of medicines written by the doctor in her or his handwriting, that is, a handwritten prescription.

In the brute-force approach, every word received from the Vision API is a candidate word and is matched with all the medicines' names stored in the database. For a large database, this matching is computationally expensive and could potentially increase the rate of false positives.

To give an idea of the possible number of matchings that have to be performed, we present a few numbers here, the details, however, are explained in Section 3. The average number of words received from the Vision API for a typical prescription is about 200. The number of SKUs that we have in our database is about 95 K. Therefore, the average number of evaluations of matches that have to be performed for a single prescription is about 19 million.

Thus, it is easy to imagine that the brute-force approach is a high latency and possibly low precision approach. However, since all words from the OCR are duly considered, a potential advantage of the brute-force approach could be a good recall, that is, a higher rate of true positives. These are some of the hypotheses that we will explore in Section 3.

For printed prescriptions, the brute-force technique entails exact matching of the OCR words with the SKU names. However, owing to slightly inferior output from the OCR for handwritten prescriptions, the brute-force approach for handwritten prescriptions uses fuzzy instead of exact matching. The brute-force approach will be the benchmark against which we will evaluate our algorithms.

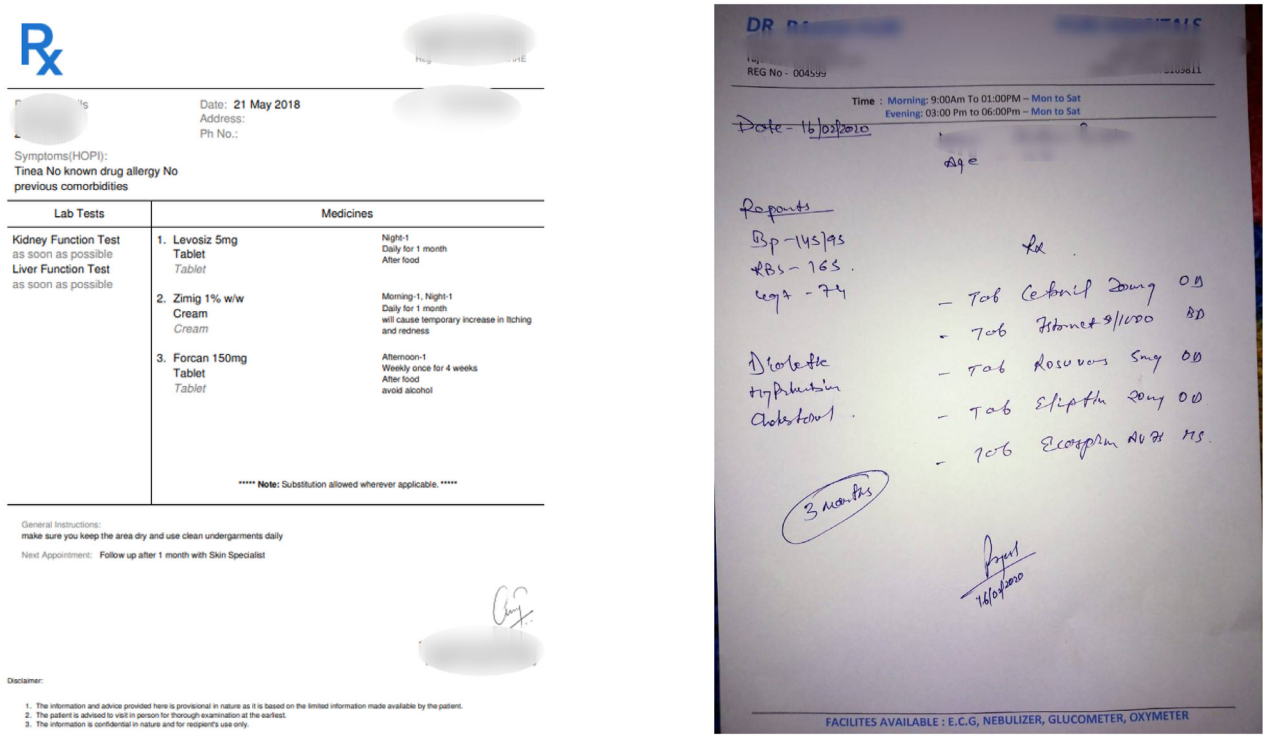
Therefore, as the brute-force approach is expensive and potentially inaccurate, we endeavoured to develop specialized algorithms which could mitigate the problem of latency and, at the same time, produce more accurate predictions. The foundations of these algorithms are built on some patterns and properties that prescriptions, in general, tend to possess. Before describing our algorithms, we discuss the patterns that we observed from a large corpus of prescriptions.

2.3. Some observations from prescriptions

To explain our methodology, we take the help of two images – a printed and a handwritten prescription – as representations. These images are presented in Fig. 2.

While developing our methodology, we manually perused several prescriptions to understand the nature of a prescription's content and structure. These observations propelled us to develop our algorithms. We describe below some of the observations that we made:

1. Words such as *Tablet*, *Injection*, *twice*, *Cap*, *Tab* are quite common in the area of the prescription where the medicines are written. The number of such words is limited and a list of such words can be compiled.
2. If multiple medicines are prescribed, they are generally written in a columnar pattern, that is, the x-co-ordinates of the positions at which the medicine names begin are almost equal. Also, the corresponding y-coordinates only



(a) A printed prescription

(b) A handwritten prescription

Fig. 2. Representational images of printed and handwritten prescriptions that were uploaded on our platform. PII information has been masked for privacy reasons.

have minor shifts for each medicine. Hence, it is possible to use the co-ordinates of the medicines' names to form a bounding box or a cluster around the area where all the medicines are written.

3. Generally, names of medicines do not contain English dictionary words. In fact, in an empirical study on our company's database (which contains more than 95 K medicines' names), we found that about 98% of the names do not contain any dictionary words.
4. A handwritten prescription is generally a mixture of printed and handwritten text with medicines' names falling in the latter category.
5. The names of the medicines are generally written with the first letter in capital.
6. The line in the prescriptions where the name of the medicine is mentioned, the dosage and frequency of the dose are also mentioned.
7. The Vision API generally predicts the first letter of the word correctly. Furthermore, the lengths of the actual and the predicted words are equal and the errors, if any, are at the character level. For example, 'a' might get predicted as 'e' but will generally not get predicted as 'ae'.
8. Medicines' names, generally, contain at least four letters. For our SKU database, we found that only 2% of the medicines' names contain fewer than four letters.
9. Printed text, in general, is written more compactly than handwritten text. This can be expressed mathematically by saying that the ratio of the area occupied by a word to the number of letters it contains is lower for printed text than for handwritten text.

We shall refer to these observations in the following sections where we describe the algorithms that we developed for both — printed and handwritten prescriptions.

2.4. Proposed solution for printed prescriptions: C–Cube algorithm

Fig. 2(a) gives an example of a printed prescription. We now explain the *C-cube algorithm*, which we developed for the digitization of printed prescriptions, using this image as an aid in the explanation. We named the algorithm C–Cube as it consists of three steps for shrinking the search space: Capture, Cluster, and Clean. These steps are explained in detail below.

Phase 1: Capture

The premise of $C - cube$ algorithm rests on two facts: medicines in a prescription are generally clustered in one area and, secondly, the medicines' names are intertwined with some common English words such as tablet, daily, once.

In the capture phase, we compile all the words (that are received from the OCR) that match exactly with the words contained in a pre-defined list, say, the *common words* list. The (x, y) co-ordinates of these words (as received from the Vision API) are also compiled.

The common words list consists of about forty words. We describe a few of them below:

1. **Units** such as *ml, mg, w/w, %, mcg,*
2. **Time** such as *afternoon, night, morning, day, month, week, evening,*
3. **Frequency** such as *once, twice, thrice,*
4. **Formulation Type** such as *tablet, cream, drops, ointment, gel, syrup,*
5. **Miscellaneous Words** such as bottle, food, breakfast, lunch, meal.

The capture step is shown in Fig. 3.

As mentioned in Observation 1 in the previous section, the words contained in the common words list are commonly found to be interspersed with the medicines' names in a prescription.

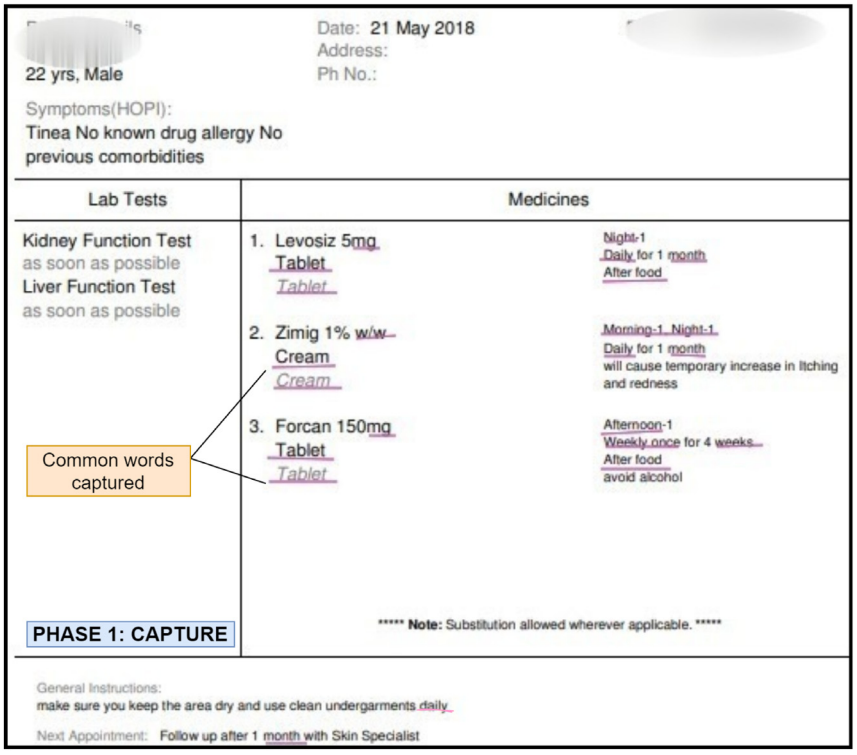


Fig. 3. Phase 1: Capture words of interest.

Therefore, if a cluster using the (x, y) coordinates of such words is formed, it is expected to encapsulate the medicines' names. This is what we accomplish in the next phase.

Phase 2: Cluster

The goals of the cluster phase are, firstly, to create a *cluster* around the area which potentially contains all the medicines' names in the prescription, and secondly, to remove outliers that lie outside this cluster.

As the printed medicines' names are mostly aligned vertically (as explained in Observation 2 previously), almost all of them lie within this cluster.

This cluster is formed with the help of the following steps:

1. Let the number of captured words be n_c. Then, the x co-ordinate of every captured word, say, x_i, $i = 1, \ldots, n_c$ is considered to be a realization of the random variable X with mean, μ_X and standard deviation, σ_x. Then, $\hat{\mu}_x = \frac{1}{n_c}\Sigma_{i=1}^{n_c} x_i = \bar{x}$ and $\hat{\sigma}_x = \sqrt{\frac{1}{n_c}\Sigma_{i=1}^{n_c}(x_i - \bar{x})^2}$. Similarly, compute $\hat{\mu}_y$ and $\hat{\sigma}_y$ for every y co-ordinate, y_i.
2. Z-scores for all x and y co-ordinates are calculated, that is, $Z_{xi} = \dfrac{x_i - \hat{\mu}_x}{\hat{\sigma}_x}$ and $Z_{yi} = \dfrac{y_i - \hat{\mu}_y}{\hat{\sigma}_y}$ for all $i = 1, \ldots, n_c$.
3. The ith word, $i = 1, \ldots, n_c$, is considered a part of the final cluster if $(|Z_{xi}| < 1.8) \cap \big(|Z_{yi}| < 1.8\big)$, that is, absolute values of Z-scores of both co-ordinates of the word are less than 1.8, which is a threshold value found through experimental tuning. Words whose co-ordinates do not satisfy the above condition are treated as outliers and are dropped from any further consideration.
4. Next, we determine the corners of the cluster formed by the words found in the previous step. For these words, we compute $\max(x)$, $\min(x)$, $\max(y)$, and $\min(y)$, where the functions $\max(.)$ and $\min(.)$ are applied over only those words that are part of the cluster formed in the previous step.

   Then, the four corners of the cluster are:
   $(\min(x), \min(y))$ $(\max(x), \min(y))$
   $(\min(x), \max(y))$ $(\max(x), \max(y))$.
5. The cluster formed in the above step is then used to short-list the words for the next phase. All words (returned by the OCR) which have either the top-left or the bottom-right co-ordinate pair falling within the above four co-ordinate pairs (corners of the cluster) are considered for the next phase.

The cluster step is shown in Fig. 4.

Finally, the list of words obtained from the above steps is then passed on to the final phase.

Phase 3: Clean

As most medicines' names are not English dictionary words (refer to Observation 3), all English dictionary words, numbers and words belonging to the common words list are eliminated from the cluster created in the previous phase to clean it.

The clean step is illustrated in Fig. 5.

The remaining words are the *candidate words*, and are all potentially medicines' names. We observed that, for printed documents, Google Vision OCR returns text with very high confidence score, in general. Therefore, matching these candidate words with the SKU database was found to be not required. Thus, these candidate words are the final output from the algorithm and the number of candidate words is the number of predicted medicines found in the prescription.

Furthermore, the fact that the algorithm can operate independently without the need of an external database, makes it suitable for discovering new medicines in the prescription, which may not even be present in the database of existing medicines' names.

For handwritten prescriptions, we found that the C-cube algorithm did not perform adequately. Due to lower confidence of the Vision API for handwritten text, the Capture phase was not effective in capturing the words of interest. Therefore, we developed a different methodology, specifically for handwritten prescriptions. This is described in the next section.

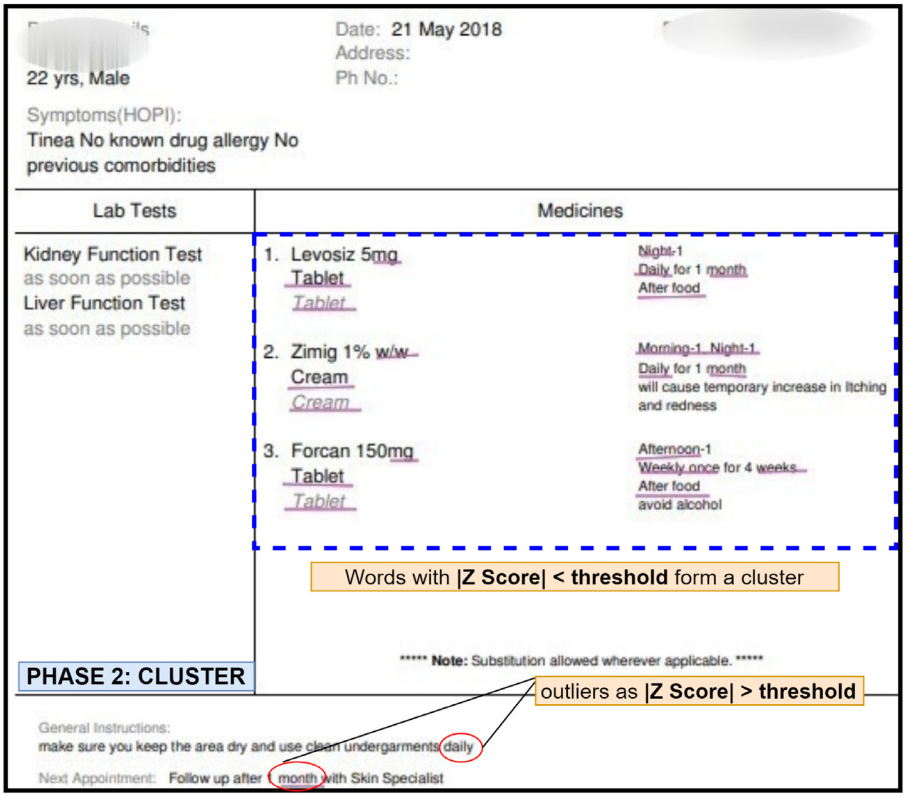


Fig. 4. Phase 2: Form a cluster which potentially contains the names of the medicines.

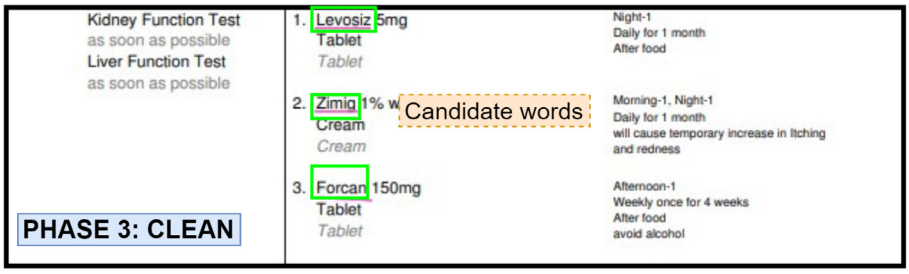


Fig. 5. Phase 3: Clean the cluster to extract the names of the medicines. Three names were found in this prescription.

2.5. Proposed solution for handwritten prescriptions: 3-Step Filtering Algorithm

Fig. 2(b) gives an example of a handwritten prescription. We shall explain the *3-Step Filtering algorithm* that we developed for the digitization of handwritten prescriptions using this image as an aid in the explanation.

We named it 3-Step Filtering as it consists of three different filtering layers to filter irrelevant text from Vision API's TextAnnotation section to detect medicines present in the prescription.

These steps are explained in detail below.

Filter 1

As a handwritten prescription is a mixture of both printed and handwritten text, this filter strives for the detection and separation of these two types. The input is a list of all words that are received from the Vision API. The output from this filter is a list of words that are predicted to be handwritten which is then passed on to the next filter. The words not predicted as handwritten are dropped from any further consideration.

As mentioned in Observation 9 in Section 2.3, the ratio of the area occupied by a word to the number of letters it contains is lower for printed text than for handwritten text. This observation is used to devise the logic of Filter 1, which entails the following steps:

1. If N_w is the number of words received from the OCR, we define $R_i = A_i/L_i$ as the *spread ratio* of the ith word, where A_i is the area occupied by the ith word and L_i is the number of letters in that word, $i = 1, \ldots, N_w$.
2. Calculate $\bar{R} = \frac{1}{N_w} \Sigma_{i=1}^{N_w} R_i$ as the average spread ratio for the prescription image.
3. The ith word is predicted to be handwritten and passed on to the next step if $R_i \geq C \times \bar{R}$, $i = 1, \ldots, N_w$, where C is a constant whose value was determined through experimental tuning.

This step is illustrated in Fig. 6.

Since printed words are expected to be packed more tightly and consequently have a lower spread ratio, the condition specified in the last step ensures that they are excluded from further consideration.

The value of the constant C was determined to be 0.80. The constant is needed to slightly lower the threshold since there

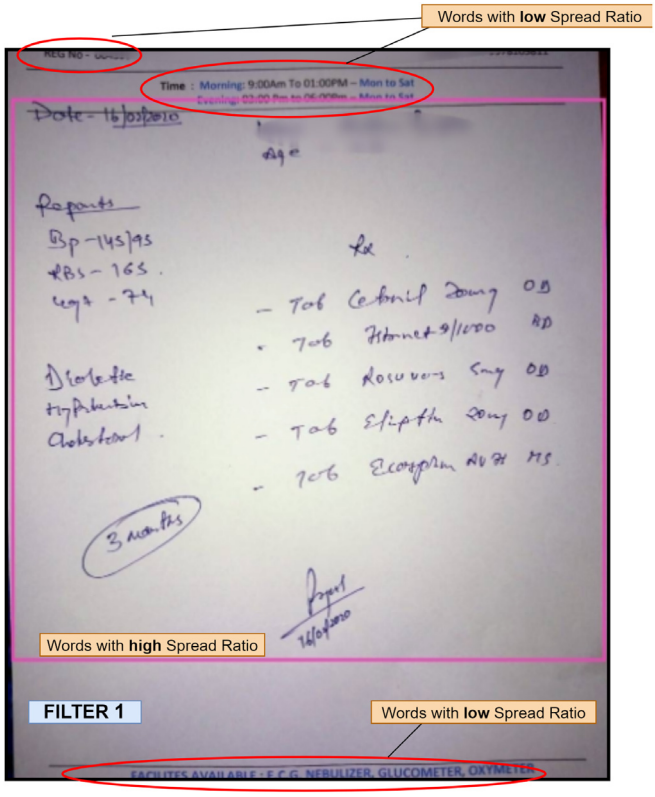


Fig. 6. Filter 1: Extract the handwritten words.

would also be words from the letterhead of a prescription which, generally, have a larger font size. Since the spread ratio of these words would be much larger than the ratio for other words, the average spread ratio, $\bar{R}$ would get inflated because of such words. Therefore, to neutralize the effect of this phenomenon, the constant $C = 0.8$ was found necessary.

Next, the words predicted as handwritten are passed to Filter 2.

Filter 2

Let us recap two observations made in Section 2.3:

Observation 3 stated that medicines' names generally do not contain English dictionary words and Observation 8 stated that medicines generally contain at least four letters.

The objective of this filter is simply to ensure that words that are either contained in the English dictionary or are less than four letters are dropped from any further consideration.

This step is illustrated in Fig. 7.

The words that remain after the above two sets of words have been removed are referred to as candidate words and are passed on to the next step.

Filter 3

In the final step, the candidate words obtained from Filter 2 are fuzzy matched with a database containing all medicines' names.

In our case, our medicines' database consists of more than 95 K names, as mentioned previously. The application of Filters 1 and 2 reduces the computational burden substantially by shrinking the search space of words.

We cannot use exact matching in this case as the results from the Vision API for handwritten text are seldom perfect and have some degree of error. Through experimental trial and error we found that fuzzy matching with a fuzzy score threshold of 70%

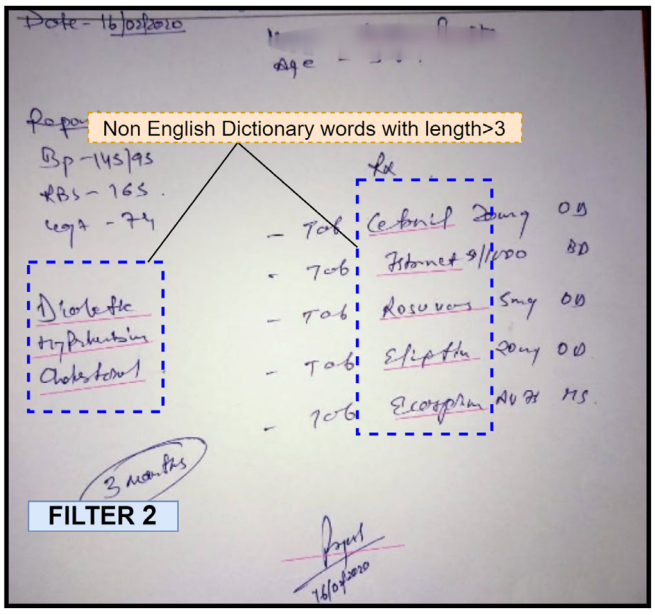


Fig. 7. Filter 2: Only non-English dictionary words and having at least four letters are passed to the next step, other words are dropped from any further consideration.

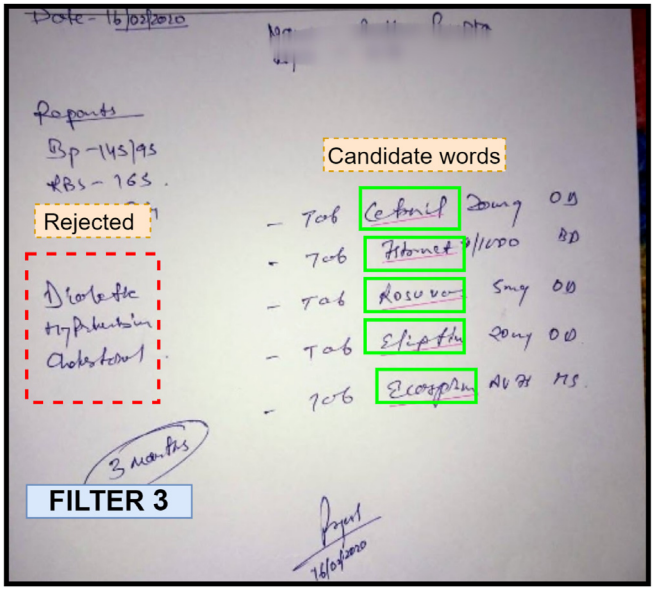


Fig. 8. Filter 3: Fuzzy matching of candidate words with a database of medicines' names.

provided the best balance between precision and recall. Levenshtein distance is used for fuzzy matching, e.g., Konstantinidis (2007). We present the results later, in Section 3.

This step is illustrated in Fig. 8.

For our application, since our medicines database contains more than 95 K names, we have added a couple of checks to increase the speed and reduce the rate of false positives, as discussed below.

1. Fuzzy matching is done only with SKU names whose first letter is the same as the candidate word's first letter. This follows from Observation 7 mentioned in Section 2.3. Other SKUs whose first letters are different from the candidate word's, are not considered at all.

2. Again, as mentioned in Observation 7, lengths of the actual medicine name and the candidate word are expected to be similar. This fact is leveraged in reducing the search space further. Only SKUs that have word lengths that are within the interval of ± 2 letters of the candidate word's length are considered for fuzzy matching.

The output from Filter 3 is a list of the predicted medicines' names which had a positive fuzzy match outcome. It may be noted that the predicted list could be null if a positive match was not obtained for any candidate word. Furthermore, there could be cases in which even the set of candidate words is null when no appropriate words get extracted by Filter 2. Similarly, the algorithm could predict more medicines than actually present in the prescription, a case of having false positives. We discuss these points in detail in Section 3.

2.6. Implementation details

Before discussing the results, we present a comparison of the three approaches that we have discussed so far in terms of their implementation: Brute-force, C-cube and 3-Step filtering algorithms.

Fig. 9 gives an overview of the three approaches.

All three approaches depend on the output of the OCR, which in our case, is the Google Vision API. In the brute-force approach, each of these words is treated as a candidate word and is matched with the names in the database. For handwritten prescriptions, fuzzy matching is performed whereas for printed ones, owing to a better quality of output from the OCR, an exact match is performed.

For handwritten prescriptions, the 3-Step filtering algorithm follows the same structure except that the filters of the algorithm reduce the search space considerably and this is expected to increase the turnaround time per prescription and reduce the rate of false positives.

The C-cube algorithm for printed prescriptions is distinct from the above two approaches by being not dependent on an external database for matching of candidate words. As mentioned before, this property is advantageous as it can help in digitization of even newer medicines that may not have been yet registered in any database. Furthermore, since no matching is performed, C-cube is expected to be the fastest algorithm as compared to the other approaches.

Next, we present the performance of these three approaches.

3. Results

3.1. Experimental setup

We evaluate the two algorithms over a corpus of 10,176 prescriptions of which 5,176 prescriptions are printed (for evaluation of C-cube algorithm) and 5,000 prescriptions are handwritten (for evaluation of 3-Step Filtering algorithm).

Fig. 10 presents the percentage distribution of the prescriptions according to the number of prescribed medicines.

It can be observed from the figure that for up to five medicines per prescription, there is a greater percentage of handwritten prescriptions, whereas, for more than five medicines, the converse is true. The average number of medicines in a prescription is 4.5 for printed and 4.0 for handwritten prescriptions.

3.2. Performance metrics of the algorithms

We present the metrics under three themes: Firstly, we expound how the two algorithms enable a reduction in the search space for the medicines' names. As mentioned before, the objective of the algorithms is to reduce the number of candidate words so that processing can be faster. Secondly, we present metrics that compare the throughputs of the brute force approach and our algorithms and demonstrate that our algorithms are decidedly faster and the predictions are more accurate. Finally, we present metrics that demonstrate that our algorithms are much more accurate than even the brute-force approach.

3.2.1. Shrinkage of the search space

The distributions of the words for printed and handwritten prescriptions are presented in Table 1. We draw comparison between the distributions of the words obtained from the OCR and the distribution of the candidate words.

The *shrink ratio* of a prescription is defined as $\left(\frac{N_w - N_c}{N_w}\right) \times 100$, where N_w is the number of words obtained from the OCR and N_c is the number of candidate words received from our algorithm.

As presented in the table, while the average number of words for handwritten prescriptions is 162.9, the average number of candidate words is just 12.9 with the average shrink ratio of 91.1%. This shrinkage reduces the computational burden on account of the fuzzy matching, as will be reflected in the throughput metrics discussed later.

For handwritten prescriptions, the distribution is for the words that are obtained from Filter 2, that is, the candidate words for the 3-Step filtering algorithm. These words are thereafter fuzzy matched with the SKU names as part of Filter 3. As mentioned before, for printed prescriptions, the candidate words obtained from the C-cube algorithm are the final output. The average number of the candidate words 3.7 is, therefore, close to the average number of medicines per prescription which, as mentioned before, is 4.5.

It can be seen in the table that the number of candidate words returned from the two algorithms can sometimes be zero as well. In such cases, the final output is null, that is, no medicine name is predicted.

Thus, by applying the two algorithms, significant computational burden can be avoided as compared to the brute-force approach. This is reflected in the throughput metrics, as discussed next.

3.2.2. Throughput metrics

The two algorithms and the brute-force approach were evaluated on Intel® Xeon® E5-2686 v4 machine with 2.30 GHz CPU and 64 GB RAM. The throughput rates are presented in Table 2

It can be observed that the two algorithms are decidedly faster in production environment as compared to the brute-force approach. While 6,000 printed prescriptions can be processed per minute, on average, by the c-cube algorithm, the brute-force approach allows only 10.2 prescriptions per minute. Similarly, about 46.2 prescriptions can be processed per minute by the 3-step algorithm whereas by brute force, it takes 5 min, on average, to process a single prescription.

In real-life applications in a production environment, the application of the two algorithms would increase the output and lead to higher process efficiency.

Next, we describe how the two algorithms are not just faster, but also are more accurate in determination of the medicines' names in the prescriptions.

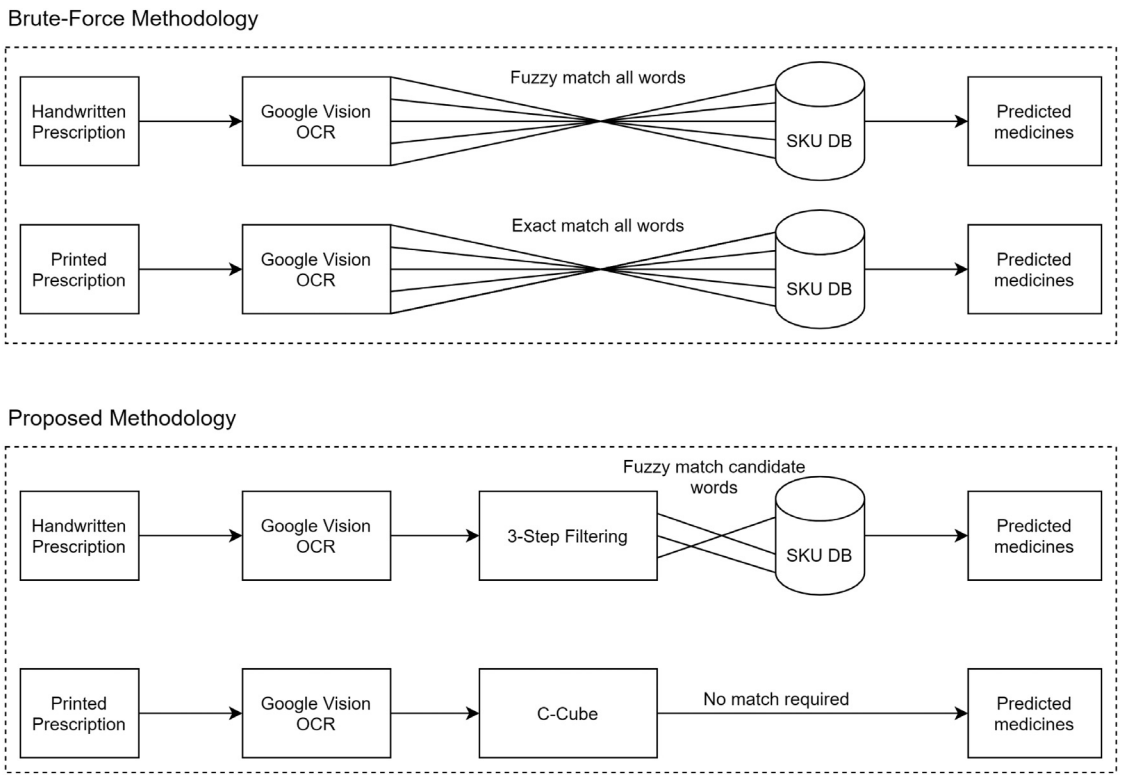


Fig. 9. Overview of the brute-force and the proposed methodologies.

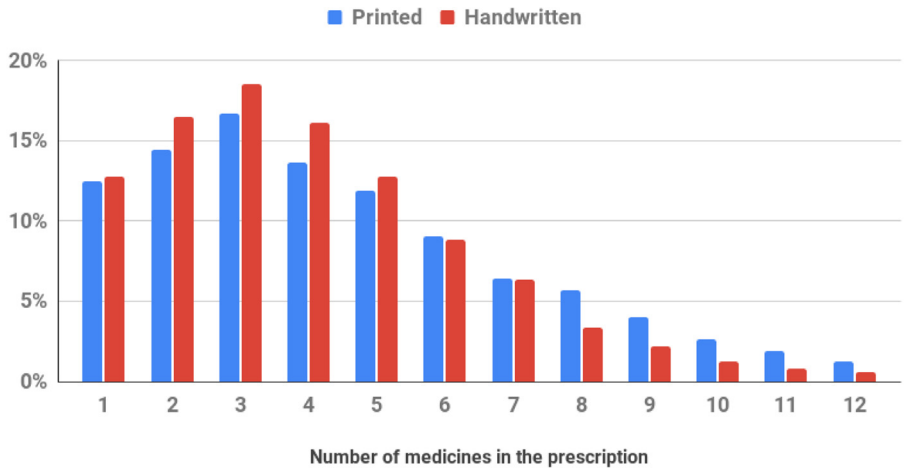


Fig. 10. Distribution of the prescriptions according to the number of prescribed medicines.

Table 1
Distributions of the number of words that are received from the OCR for the two types of prescriptions (without application of the algorithms) and after the two algorithms are applied.

Prescription type	Min	Q_1	Mean	Q_3	Max	Average shrink ratio
Distribution of the words obtained from the OCR						
Printed	7	154	241.1	302	877	–
Handwritten	8	97	162.9	210	955	–
Distribution of the candidate words obtained from the two algorithms						
Printed	0	2	3.7	5	46	98.4%
Handwritten	0	8	12.9	17	84	91.1%

Table 2
Average number of prescriptions processed per minute using different methods.

Prescription type	Brute-force technique	Post application of our algorithms	
	Rx digitized/ minute	Rx digitized/ minute	Percentage gain
Printed	10.2	6,000	58,724%
Handwritten	0.20	46.2	22,977%

3.2.3. Accuracy metrics

We present the accuracy of the algorithms in two dimensions. Firstly, we present the metrics on an overall level where we compare the precision, recall and F1 score metrics for the different approaches. Secondly, we present the distributions of the results according to the number of medicines contained in the prescriptions.

On overall level

As mentioned earlier, the output from each of the three approaches (Brute-Force, C-cube and 3-Step Filtering) is a list of medicines' names. The number of predicted medicines' names could range from 0 to N_w. To compare the performances of these approaches, we compare the predicted list of medicines with the actual list of medicines that were prescribed. Note that this is just for performance measurement — none of the three approaches depend on the actual list of the prescribed medicines.

Let us define:

true positives (TP) as the number of medicine names in the actual prescription that also appear in the predicted list of medicine names,

false positives (FP) as the number of medicine names that are not present in the actual prescription but appear in the predicted list, and

false negatives (FN) as the number of medicine names present in the prescription but not present in the predicted list.

Then, $\text{Precision} = \frac{TP}{(TP + FP)}$, $\text{Recall} = \frac{TP}{(TP + FN)}$, and $\text{F-Score} = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}$.

The above metrics are computed for the brute-force approach (both printed and handwritten prescriptions), C-cube (for printed prescriptions) and 3-Step Filtering (for handwritten prescriptions), and are presented in Table 3.

It can be seen from the table that the application of C–Cube and 3-Step Filtering algorithms lead to a substantial increase in the precision as compared to the brute-force approach. While precision for the printed prescriptions trebles, for handwritten prescriptions it becomes more than ten times. Although these gains come at a slight cost of recall, the substantially improved F-score (which is a harmonic mean of precision and recall) emphasizes on the improved accuracy standards that our algorithms are able to achieve.

As shown in Fig. 9, in the case of the brute-force approach, the words received from the OCR are fuzzy matched with the SKUs database. Since the average number of words obtained from the OCR is quite large – about 163 (Table 1) – this leads to a low value of precision. The effectiveness of the 3-step filtering algorithm in shrinking the average number of candidate words to about 13 enables it to achieve a much higher precision. While there is some scope of improvement of the F-score for handwritten prescriptions, it is still considerably better than the one obtained from the brute-force approach.

On a prescription level

As shown in Table 3, the recall for the two algorithms is 70% and 34.7%. This means that, on average, about 70% of the prescribed medicines in the printed and about 35% of the prescribed medicines in the handwritten prescriptions are correctly recovered by the two algorithms.

Let us now explore the recall of the two algorithms for prescriptions containing different numbers of prescribed medicines. Fig. 11 presents the percentage distributions of prescriptions according to the number of medicines contained in them and the number of medicines correctly recovered.

Next, in Fig. 12, we present data about average number of medicines that were correctly recovered, along with 95% confidence intervals, for the two algorithms.

It can be seen from Fig. 11(a) that the percentage of prescriptions with perfect match rate, that is, when all the medicines are correctly recovered decreases with the number of medicines in the prescription, which is intuitive as the joint likelihood diminishes with the increase in the number of trials. However, for printed prescriptions, even when the number of prescribed medicines is as large as eight, all the medicines are correctly recovered for about 20% of the prescriptions.

Similarly, the percentage of prescriptions in which at least one medicine is correctly recovered increases with an increase in the number of prescribed medicines, as can be seen in Fig. 11(b). For prescriptions containing a large number of medicines (more than eight per prescription), almost 100% in the case of printed prescriptions and about 80% in the case of handwritten prescriptions are such that at least one prescription is successfully recovered.

Therefore, by demonstrating the faster throughput and accuracy metrics, we establish the superiority of our algorithms over the brute-force approach.

3.3. Failure analysis

The metrics that we have presented in the preceding section demonstrate the effectiveness of the two algorithms. There are, however, many instances when the algorithms fail to correctly identify even a single medicine name in the prescription image. Fig. 13 shows the distribution of the prescriptions in which the algorithms could not correctly identify any medicine name.

For prescriptions that only contain name of only one medicine, the C-cube algorithm failed to detect that medicine name in about 27% whereas for handwritten prescriptions, the 3-Step filtering algorithm failed in 64% of the prescriptions. As expected, the percentages of complete failures decrease as the number of medicines in the prescription increases.

It is pertinent to analyse the reasons behind these failures as that might help us in taking corrective actions for future prescriptions. Before discussing specific examples that lead to these failures, let us discuss the two broad causes behind these failures.

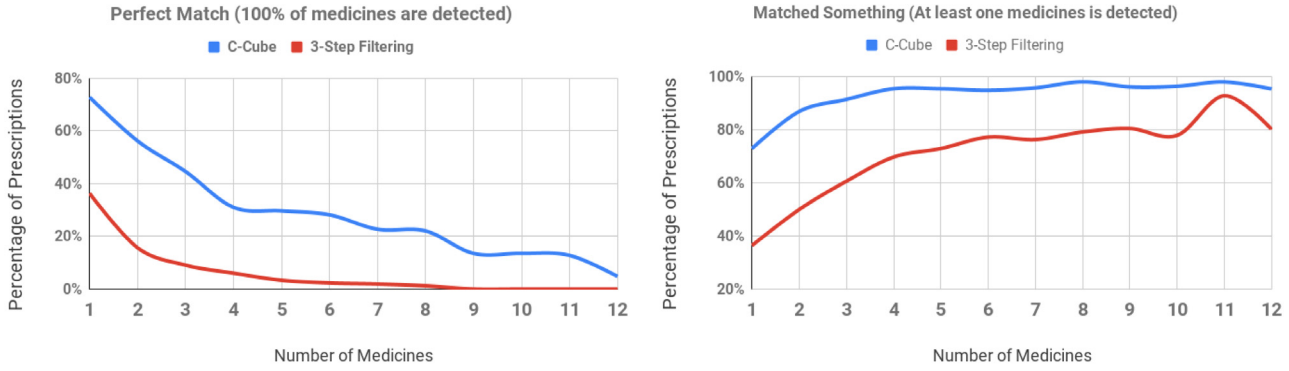
Type A errors: These are failures due to the inability of the OCR (the Vision API in our case) to correctly identify the characters. This mainly happens when either the handwriting is such that is difficult to read even by the human eye or that quality of the uploaded image is very poor and there are issues such as blurriness and creases in the paper. Or poor formatting of the input text leading to erroneous interpretation such as detecting space between two words when there was none and treating them as two separate words instead of one single word.

Type B errors: These are failures due to the limitations of our algorithms. For example, violation of some of the assumptions that we made.

Let us now illustrate a few specific examples on how the above mentioned reasons lead to failure of the algorithms to detect even a single medicine name correctly in the given prescription.

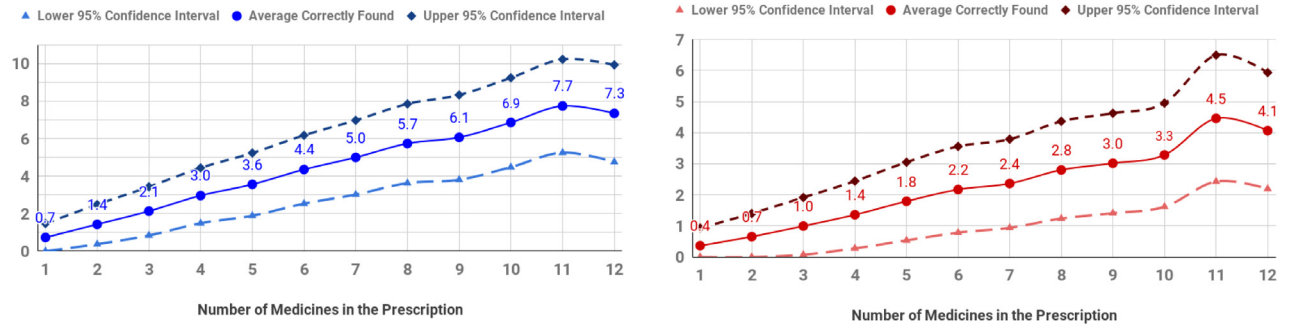
Table 3
Accuracy metrics for the three approaches: Brute-force, C-cube (printed prescriptions) and 3-step filtering (handwritten prescriptions).

Prescription type	Brute-force technique			Post application of our algorithms		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Printed	29.9%	82.7%	41.8%	91.6%	70.0%	79.4%
Handwritten	0.2%	62.0%	0.3%	21.1%	34.7%	26.2%



(a) All the prescribed medicines are recovered. (b) At least one medicine is recovered.

Fig. 11. Percentage distributions of prescriptions according to the number of medicines contained in them and the number of medicines correctly recovered.



(a) Handwritten Prescriptions (b) Printed Prescriptions

Fig. 12. Average number of medicines that are correctly recovered, along with 95% confidence intervals.

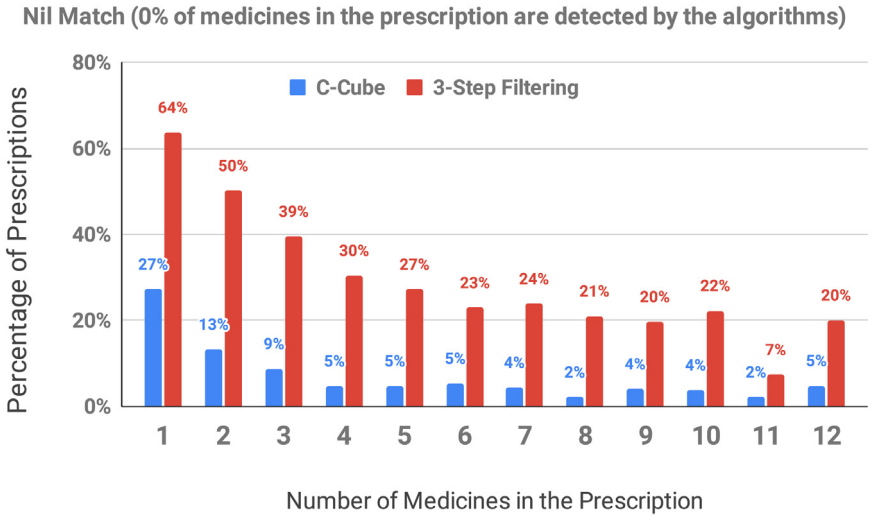
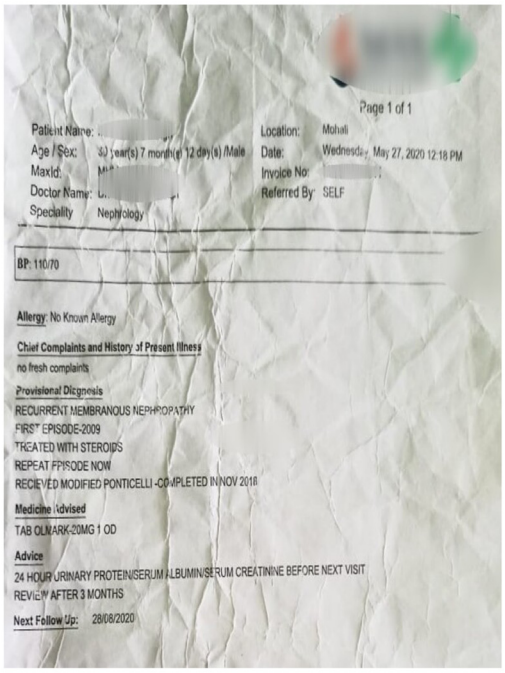
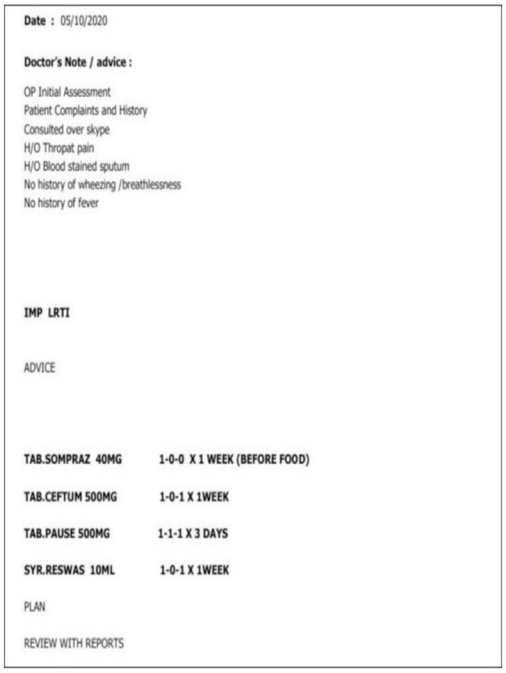


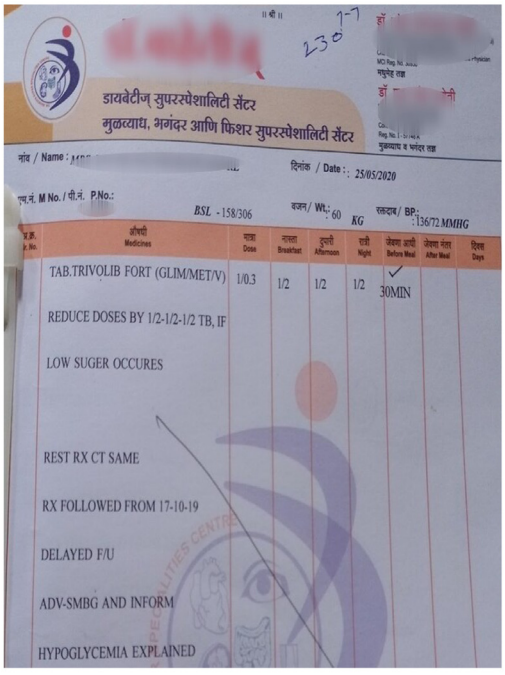
Fig. 13. Filter 3: Fuzzy matching of candidate words with a database of medicines' names.



(a) Poor image quality (In this case, the page is crumpled)



(b) Irregular spacing (TAB.X has no space so it is detected as one word, where X is medicine name)



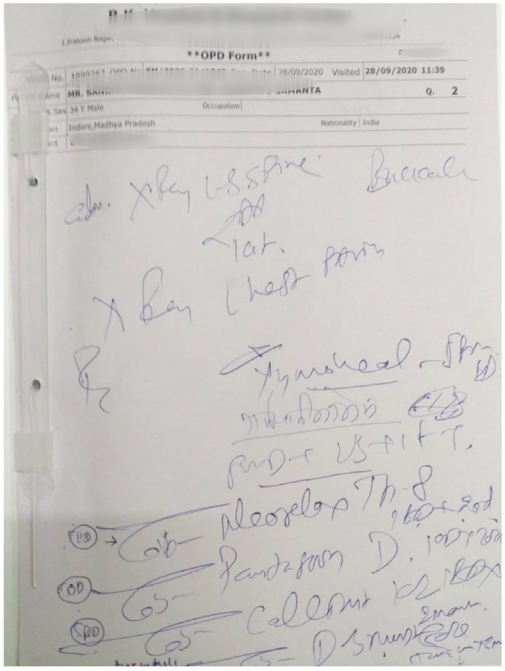
(c) Cluster phase failed as the common words found were not enough for clustering (see Section 2.4)

(d) The prescribed medicine's name, *Pause*, is an English dictionary word

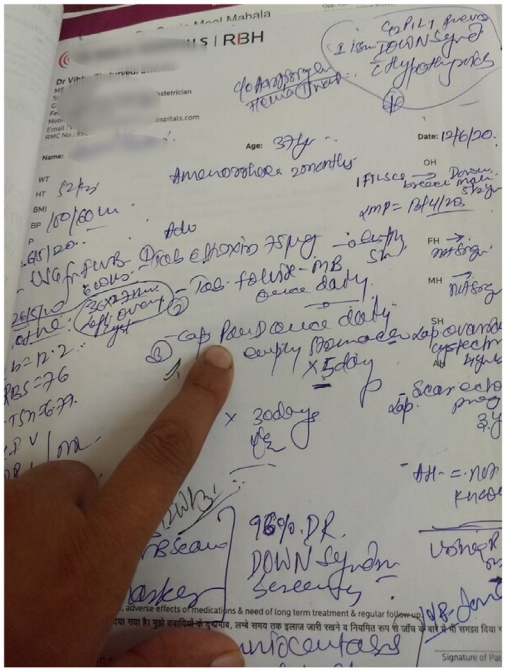
Fig. 14. Four examples where the C-cube algorithm failed to identify a single medicine name in the printed prescription.

We first discuss the C-cube algorithm for printed prescriptions. Fig. 14 presents four examples when the algorithm failed. We observed four broad themes behind the failure of the algorithm:

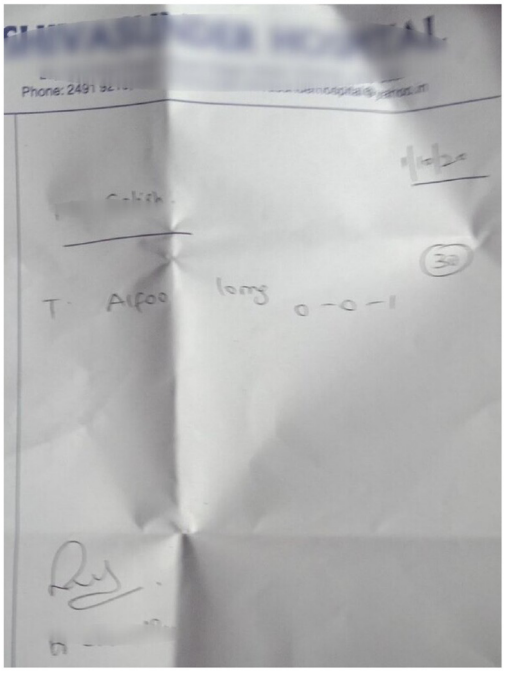
- Poor image quality (Type A error). For example, the image is blurred or the page is crumpled or obscured. Fig. 14(a) illustrates this point.
- Poor text formatting (Type A error). Words are incorrectly read because of poor text placement. For example, detecting space between two words when there was none and treating them as two separate words instead of one single word. See Fig. 14(b) for an example of this type.
- Failure of the *cluster* phase of the algorithm (Type B error). For sparse prescriptions, that is when only one or two



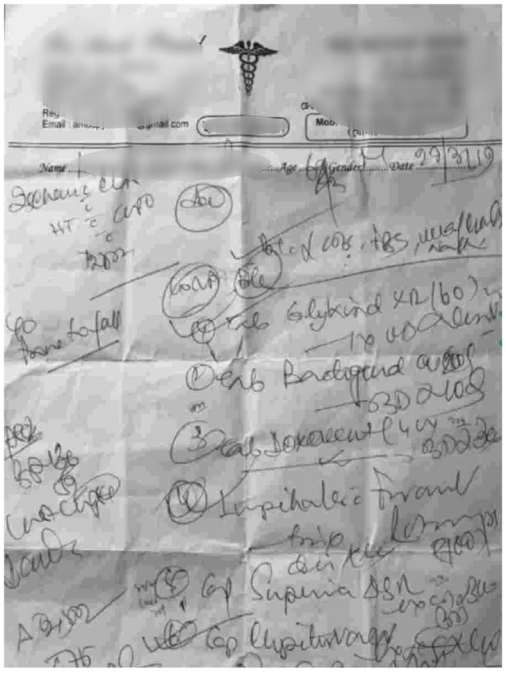
(a) Cacography: There is some robustness to slightly bad handwriting but only until a point



(b) Poor image quality: In this case, the uploaded image is partially obscured



(c) Filter 1 failure: printed text in the letterhead dominates the handwritten text (see Section 2.5)



(d) Bad formatting of the handwritten text: In this case, cluttered text leads to failure of the algorithm

Fig. 15. Four examples where the 3-Step filtering algorithm failed to identify a single medicine name in the printed prescription.

medicines are mentioned in the entire image, the cluster phase of the algorithm would have a tendency to fail. In Fig. 13, it can be seen that the decline in percentage failures for the C-cube algorithm with the increase in the number of medicines is much sharper than the decline for the 3-step filtering algorithm. Fig. 14(c) illustrates such a prescription which led to the failure of the algorithm.

- Other reasons (Both Type A and B errors). Factors such as medicines' names being composed of English dictionary words, prescriptions in vernacular languages, medicines' names not mentioned in full but instead abbreviations are used, also contribute to the failure of the algorithm to successfully detect the medicine name. Fig. 14(d) illustrates a prescription that contains a medicine name which is an

English language word. As mentioned in point 3 of Section 2.3, one of the assumptions behind the algorithms is that names of medicines do not contain English dictionary words. Therefore, for this particular prescription, the algorithm is unsuccessful.

Next, we discuss the 3-Step filtering algorithm for handwritten prescriptions. Fig. 15 presents four examples when the algorithm failed. Here also, we observed four broad themes behind the failure of the algorithm:

- Cacography (Type A error). This is one of the most prominent and intuitive reasons behind the failure of the algorithm. We found that while slightly bad handwriting might still be manageable, the robustness is quite low. Fig. 15(a) illustrates a prescription for which the algorithm failed because of the cacography.
- Poor image quality (Type A error). For example, the image is blurred or the page is crumpled or obscured. Fig. 15(b) presents an example of such an image that led to the failure of the algorithm.
- Filter I failure (Type B error). As mentioned in Section 2.5, the role of the first filter is to separate the printed words from the handwritten words in the image. The underlying logic is to compute the average spread ratio and compare it with the spread ratios of individual words (as mentioned in point 3 of Section 2.5). In the case of very few medicine words, the average spread ratio will be dominated by words contained in the letterhead of the prescription, thus inflating it. Because of this, the condition mentioned in point 3 will not be met by any word and, thus, no word will be classified as handwritten. This would result in the termination of the algorithm as no words are available for further processing in subsequent filters. Fig. 15(c) presents an example of a failed instance because of this reason. Filter I can also fail when the handwriting of the doctor is too small.
- Other reasons (Both Type A and B errors). For example, the handwritten text is cluttered and lacks a coherent format. Or excessive spacing between the characters leading to erroneous interpretation of being multiple words. Fig. 15(d) presents an example of such a prescription image.

One of the aforementioned causes — poor quality of the uploaded image, is avoidable. We, therefore, encourage our customers to upload high quality and clearly visible images of their prescriptions at the time of placing an order on our platform.

4. Discussion

In this paper, we presented algorithms for faster and more accurate extraction of medicines' names from the prescriptions. The low latencies of our algorithms make them suitable for use in a production environment wherein the results are required on a real-time basis. Furthermore, our algorithms can be used as plug-and-play since neither there is any requirement of training on any dataset nor is any labelling or annotation that is required to be done.

The implementation of the two algorithms within our systems is also compliant from a data security point of view. Within our systems, customer and patient information is stored in a secure environment, in full compliance with the government laws and regulations. Regarding uploading of the prescriptions' images to Google Vision API, the documentation, Google (2021b), clarifies that (a) the uploaded images are not used in any way for improvement of the service, (b) uploaded images are not saved and are deleted immediately once processing is completed, and (c) the images and the data are not shared with any third parties, except as necessary to provide the Vision API service (under appropriate security and confidentiality contractual obligations).

As mentioned in Section 2.5, the threshold for the fuzzy score was chosen to be 70% as it resulted in the best balance between precision and recall. In general, a lower threshold results in a higher recall but lower precision, and vice versa. If there is a specific requirement of achieving a higher precision (recall), the threshold can be increased (decreased) accordingly.

The brute-force approach has a recall that is better than both the algorithms, the precision, however quite poor. Latency of the brute-force approach is also quite high. Therefore, even with a high recall, we do not think that the brute-force approach will be preferred over our algorithms in any practical application. Our choice of the performance metric, the F-score, which is the harmonic mean of precision and recall, ensures a balance between the two metrics and therefore, is an ideal metric for such systems. As discussed in Section 3, our algorithms increase the F-scores by 90% and 8,600% over the brute-force approach and are, therefore, decidedly superior in accuracy with the additional advantage of substantially higher processing speeds.

The precision of the C-cube algorithm is about 92% which is already quite high. Also, there is no requirement of matching with any SKUs database which makes this algorithm independent and bestows it with the ability to discover new medicines' names that may not have been registered in a database. If there is a need to increase the precision of the C-cube algorithm further, the candidate words found by the algorithm can be exactly matched with an SKU database. This might increase the precision marginally but could come at the cost of a slight reduction in the recall and increased latency.

As discussed in Section 2.4, we used Z-scores to decide whether a word is a part of the predicted medicines' area cluster. We also explored other clustering algorithms such as k-means and DBSCAN, Ester et al. (1996). However both these methods performed inadequately. The problem with the k-means algorithm was excessive sensitivity to extreme values (outliers). The problem with DBSCAN was the inability to optimize the radius hyperparameter. As a typical prescription shows large variability in how different words and lines are arranged, it was difficult to choose the right value of the radius parameter for the DBSCAN algorithm for every prescription.

The 3-Step filtering algorithm, as discussed in Section 2.5, relies upon fuzzy matching using Levenshtein distance. Other tree-based algorithms such as BK Tree and FQ Trees, Baeza-Yates et al. (1994) could be explored for superior fuzzy match in terms of reduced latency and increased recall in case of large database of SKUs. However, no performance metrics have been published for large datasets for these algorithms so it is difficult to predict their performance.

A limitation of our algorithms is the dependence on an external OCR. While we have relied upon the OCR output returned from the Google® Vision API, other equivalent alternatives can also be used. For example, Microsoft® provides a service called Computer Vision that also returns the words and their corresponding co-ordinates, Microsoft (2021). However, currently, it is not possible to replace these OCRs with an open-source equivalent. This might limit the uptake of our algorithms in cost-sensitive applications.

As to improvement and enhancement to our algorithms, we plan to investigate the following ideas. Firstly, the current version of our algorithms only predict medicines' names. In a real-world application, it will be desirable to have accompanying prediction scores. These scores could be evaluated against a threshold score and could lead to a further reduction in the rate of false positives. The confidence scores returned by the Vision API could be an important factor in the computation of this prediction score.

Secondly, the algorithms can be made amenable to inclusion of auxiliary information extracted from the current or past prescriptions, if available. For example, the speciality of the prescribing doctor is often mentioned on the prescription. This speciality can be extracted from the prescription and can provide context about the disease, which can then be used to refine the search space for the medicines' names. For example, it is unlikely for an ophthalmologist to prescribe a cardiac-related drug. Similarly, if past prescriptions of the patient are available, the combined OCR data can be helpful in increasing the performance of fuzzy matching, thereby increasing the precision and recall of the algorithms.

Thirdly, to make this a wholesome solution, extracting medicines' names only may not suffice for some applications. It may also be necessary to extract incidental information such as the dosage, frequency and duration. Since we are generally able to precisely locate the cluster which contains the medicines' names, methods can be developed which extract these additional data, which are already present in those clusters, along with the medicines' names. By adding these features to our algorithms, a complete solution which extracts the entire drug-related information from a given prescription could be developed.

Finally, while Google® Vision API is state-of-the-art, (Hosseini et al., 2017) showed that it is not immune to image noise and even minor deteriorations in the image quality can significantly reduce the accuracy of the output. Therefore, pre-processing actions such as cropping, grey-scaling and blur reduction could possibly enable better quality output. The improved quality of the output would result in candidate words which are closer to the actual text, and would invigorate both algorithms, particularly 3-Step filtering since the doctor's cacography leads to relatively poor performance. We intend to work on the above ideas in the future.

We foresee and propose the following applications of our algorithms. Firstly, we envision integration of these algorithms in a smartphone application for real-time use. This can enable patients to understand the medicines that they have been prescribed not just the current list of medicines, but also the ones that they were prescribed in the past. This can help patients with the anxiety that they might have about the treatment that they receive, Nair et al. (2002). Furthermore, by putting a text-to-speech layer over these algorithms, they can assist people with visual problems or illiterate people in understanding the names of the drugs prescribed to them, Wolf et al. (2007), Zhi-Han et al. (2017). If these names are also linked to medicines' meta-data, they can greatly improve a patient's confidence in the treatment that they are receiving, potentially leading to greater adherence to the treatment regimen, Davis et al. (2008).

Secondly, these algorithms can also be used to digitize historical prescriptions, in batch mode. A prescription is brimming with healthcare data and the cumulation of such information can help with the management of medicine history and creation of a patient profile. Longitudinal studies can be implemented on patients having a particular characteristic to forecast disease prognosis and for development of disease models, Wang et al. (2014), Caruana et al. (2015), Almario (2017) and Davenport and Kalakota (2019). Since our algorithms have very low latency, they could be integrated with hospitals' data management systems to digitize prescriptions on the fly.

Thirdly, other e-pharmacies can benefit from these algorithms since it is a statutory requirement in most parts of the world that medicines should be dispensed only after properly vetting the prescription. These algorithms can be used to assist the staff engaged in screening and data-entry of the incoming prescriptions and can, thus, lead to increased productivity of the employees.

Finally, we think that the logic behind the C-cube algorithm, wherein target words are captured and a cluster is created could be applied to other document types as well. This methodology can be applied to documents that have a fixed structure and there are a few standard keywords that accompany the words of interest. Such documents include diagnostic reports, invoices, receipts and specific legal documents.

In summary, we have presented algorithms for extracting medicines' names from both, printed as well as handwritten prescriptions. Our algorithms have low latency and high accuracy, thus making them suitable for real-time production environment as well as post-hoc digitization of health records. Such solutions can be a boon for all the stakeholders in the healthcare system by not just reducing the digitization related costs, but the data garnered through these algorithms could be of immense use in building advance warning systems to alert the community about emerging patterns before they become major health and economic risks, Agrebi and Larbi (2020), Davenport and Kalakota (2019). Aggregated and filtered data extracted by the algorithms can be used as an input in epidemiology intelligence systems to detect emergence of novel health risks, so that pre-emptive actions can be undertaken, Wilburn et al. (2019).

CRediT authorship contribution statement

Mehul Gupta: Conceptualization, Methodology, Software, Supervision, Software, Validation. **Kabir Soeny:** Data curation, Writing – original draft, Visualization, Investigation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Agrebi, S., Larbi, A., 2020. Use of artificial intelligence in infectious diseases. Artif. Intell. Precis. Health 415–438.

Almario, C.V., 2017. The effect of digital health technology on patient care and research. Gastroenterol. Hepatol. 13 (7), 437–439.

Baeza-Yates, R., Cunto, W., Manber, U., Wu, S., 1994. Proximity matching using fixed-queries trees. In: Crochemore, M., Gusfield, D. (Eds.), Combinatorial Pattern Matching. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 198–212.

Bodenreider, O., 2004. The unified medical language system (UMLS): integrating biomedical terminology. Nucleic Acids Res. 32 (90001), 267D–270.

Caruana, E.J., Roman, M., Hernández-Sánchez, J., Solli, P., 2015. Longitudinal studies. J. Thoracic Dis. 7 (11), E537–E540.

Chrimes, D., Moa, B., Zamani, H., Kuo, M., 2016. Interactive healthcare big data analytics platform under simulated performance. In: 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech). pp. 811–818.

Davenport, T., Kalakota, R., 2019. The potential for artificial intelligence in healthcare. Future Healthc. J. 6 (2), 94–98.

Davis, T.C., Federman, A.D., Bass, P.F., Jackson, R.H., Middlebrooks, M., Parker, R.M., Wolf, M.S., 2008. Improving patient understanding of prescription drug label instructions. J. Gen. Intern. Med. 24 (1), 57–62.

Doan, S., Bastarache, L., Klimkowski, S., Denny, J., Xu, H., 2009. Vanderbilt's system for medication extraction. In: Proceedings of the Third I2b2 Workshop on Challenges in Natural Language Processing for Clinical Data.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. AAAI Press, pp. 226–231.

Fajardo, L., Sorillo, N., Garlit, J., Tomines, C., Abisado, M., Imperial, J.M., Rodriguez, R., Fabito, B., 2019. Doctor's Cursive Handwriting Recognition System Using Deep Learning. pp. 1–6.

Google, 2021a. Vision AI. https://cloud.google.com/vision.

Google, 2021b. Vision AI documentation. https://cloud.google.com/vision/docs/data-usage.

Grouin, C., Deleger, L., Zweigenbaum, P., 2009. A simple rule-based medication extraction system. In: Proceedings of the Third I2b2 Workshop on Challenges in Natural Language Processing for Clinical Data.

Hamon, T., Grabar, N., 2009. Concurrent linguistic annotations for identifying medication names and the related information in discharge summaries. In: Proceedings of the Third I2b2 Workshop on Challenges in Natural Language Processing for Clinical Data.

Hosseini, H., Xiao, B., Poovendran, R., 2017. Google's cloud vision API is not robust to noise. http://arxiv.org/abs/1704.05051.

Konstantinidis, S., 2007. Computing the edit distance of a regular language. Inform. and Comput. 205 (9), 1307–1316.

Li, Z., Liu, F., Antieau, L., Cao, Y., Yu, H., 2010. Lancet: a high precision medication event extraction system for clinical text. J. Am. Med. Inform. Assoc. 17 (5), 563–567.

Lyons, R., Payne, C., McCabe, M., Fielder, C., 1998. Legibility of doctors' handwriting: quantitative comparative study. BMJ : Br. Med. J. 317, 863–864.

Medicine, I., America, C., Donaldson, M., Corrigan, J., Kohn, L., 2000. To err is human: Building a safer health system. In: Quality Chasm Series, (v. 627), National Academies Press.

Microsoft, 2021. Computer vision. https://azure.microsoft.com/en-in/services/cognitive-services/computer-vision/.

Nair, K., Dolovich, L., Cassels, A., McCormack, J., Levine, M., Gray, J., Mann, K., Burns, S., 2002. What patients want to know about their medications. Focus group study of patient and clinician perspectives. Canad. Family Phys. 48, 104–110.

Patrick, J., Li, M., 2009. A cascade approach to extracting medication events. In: Proceedings of the Australasian Language Technology Association Workshop 2009. pp. 99–103.

Solt, I., Tikk, D., 2009. Yet another rule-based approach for extracting medication information from discharge summaries. In: Proceedings of the Third I2b2 Workshop on Challenges in Natural Language Processing for Clinical Data.

Spasić, I., Sarafraz, F., Keane, J.A., Nenadić, G., 2010. Medication information extraction with linguistic pattern matching and semantic rules. J. Am. Med. Inform. Assoc. 17 (5), 532–535.

Tao, C., Filannino, M., Uzuner, Ö., 2017. Prescription extraction using CRFs and word embeddings. J. Biomed. Inform. 72, 60–66.

1mg Technologies, 2021. 1mg - online medical store and healthcare platform. https://www.1mg.com.

Uzuner, O., Solti, I., Cadag, E., 2010. Extracting medication information from clinical text. J. Am. Med. Inform. Assoc.: JAMIA 17 (5), 51–518.

Wang, X., Sontag, D., Wang, F., 2014. Unsupervised learning of disease progression models. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, pp. 85–94.

Wilburn, J., O'Connor, C., Walsh, A.L., Morgan, D., 2019. Identifying potential emerging threats through epidemic intelligence activities - looking for the needle in the haystack? Int. J. Infect. Dis. 89, 146–153.

Wolf, M.S., Davis, T.C., Shrank, W., Rapp, D.N., Bass, P.F., Connor, U.M., Clayman, M., Parker, R.M., 2007. To err is human: Patient misinterpretations of prescription drug label instructions. Patient Educ. Counsel. 67 (3), 293–300.

Yang, H., 2009. Linguistic Approach for medication extraction from medical discharge. In: Proceedings of the Third I2b2 Workshop on Challenges in Natural Language Processing for Clinical Data.

Zhi-Han, L., Yow, H.Y., Makmor-Bakry, M., 2017. Medication-handling challenges among visually impaired population. Arch. Pharm. Pract. 8, 8.