scientific reports

Check for updates

OPEN

Incremental learning algorithm for dynamic evolution of domain specific vocabulary with its stability and plasticity analysis

Mansi Jain[1], Harmeet Kaur[2], Bhavna Gupta[3]✉, Jaya Gera[1] & Vandana Kalra[4]

Domain-specific vocabulary, which is crucial in fields such as Information Retrieval and Natural Language Processing, requires continuous updates to remain effective. Incremental Learning, unlike conventional methods, updates existing knowledge without retraining from scratch. This paper presents an incremental learning algorithm for updating domain-specific vocabularies. It introduces DocLib, an archive used to capture a compact footprint of previously seen data and vocabulary terms. Task-based evaluation measures the effectiveness of the updated vocabulary by using vocabulary terms to perform a downstream task of text classification. The classification accuracy gauges the effectiveness of the vocabulary in discerning unseen documents related to the domain. Experiments illustrate that multiple incremental updates maintain vocabulary relevance without compromising its effectiveness. The proposed algorithm ensures bounded memory and processing requirements, distinguishing it from conventional approaches. Novel algorithms are introduced to assess the stability and plasticity of the proposed approach, demonstrating its ability to assimilate new knowledge while retaining old insights. The generalizability of the vocabulary is tested across datasets, achieving 97.89% accuracy in identifying domain-related data. A comparison with state-of-the-art techniques using a benchmark dataset confirms the effectiveness of the proposed approach. Importantly, this approach extends beyond classification tasks, potentially benefiting other research fields.

Keywords Unigrams, Bigrams, Incremental learning, n-grams, Text analytics, Natural language processing

A domain-specific vocabulary comprises terms that are relevant, recurrent, and representative of the domain. Knowledge of such terms can be helpful in a variety of research areas, such as Information Retrieval, Machine Learning, Natural Language Processing, Artificial Intelligence, and Natural Language Generation[1,2]. Such a vocabulary, consisting of a concise selection of relevant text features representing a domain, can lower the computational cost of many applications that work on the text. For example, domain-specific vocabulary can significantly reduce the computational cost of training a classification model by reducing feature space dimensionality[3]. Other examples include applications such as Recommender Systems, Topic Identification, Name Entity Recognition, and Sentiment Analysis[4–7]. By integrating domain-specific terms, vocabulary can also be utilized for different specialized tasks in Text Analytics and Natural Language Processing applications such as Semantic Analysis, Text Summarization, and Topic Modeling. It can also improve the comprehension and interpretation of text within specific fields such as medicine[8,9], law, finance, and technology. A domain-specific vocabulary can help to choose the right terms to generate content for a particular industry or subject area. This ensures that the content is accurate, reliable, and relevant to the target audience within a particular domain. Such vocabularies are also integral to the training and fine-tuning of Large Language Models (LLM). LLMs equipped with domain-specific terminology can generate contextually more appropriate and domain-relevant text, making them valuable tools for a wide range of applications. Domain-specific terminology is even more important for technical documents, specifications, manuals, and guidelines. This can assist in maintaining consistent and precise content. However, simply creating a vocabulary is insufficient until it is updated after encountering new data related to the domain. Continual updates ensure that the vocabulary is effective and

[1]Department of Computer Science, Shyama Prasad Mukherji College for Women, University of Delhi, Delhi, India. [2]Department of Computer Science, Hansraj College, University of Delhi, Delhi, India. [3]Department of Computer Science, Keshav Mahavidyalaya, University of Delhi, Delhi, India. [4]Department of Computer Science, Sri Guru Gobind Singh College of Commerce, University of Delhi, Delhi, India. ✉email: bgupta@keshav.du.ac.in

useful for various tasks. This work aims to present a technique that incrementally updates a domain-specific vocabulary.

Conventional machine-learning models generally assume that all training data are available before model building. Thus, the model can be trained by identifying the patterns, structures, and distributions of the entire dataset under the assumption that the data are static. In contrast, incremental learning is a paradigm in which model learning is a continuous task, as the data continue to be added. Incremental learning does not assume that all the training data are available before the learning step. Learning occurs as new samples are added, and the model adapts according to the new data. The model-building process requires scanning add-on datasets without forgetting previously learned knowledge[10–12]. Incremental learning is essential for various applications in the modern digital era, as detailed in "Related work" section. Examples include autonomous and interactive systems that behave according to input situations, such as robots and autonomous cars[13–17].

An incremental learning model addresses many challenges, such as the stability-plasticity dilemma[11], concept drift, and catastrophic forgetting[18,19]. Stability refers to a model's ability to conserve previously acquired knowledge. Plasticity is the ability of a model to acquire new knowledge about an ongoing task. Focusing only on stability may impact the learning ability of the model, whereas targeting plasticity may lead to catastrophic forgetting, implying forgetting already acquired knowledge. Therefore, an incremental model must strike a balance between the two to be effective and useful. Concept drift is a scenario in which the distribution of the add-on datasets does not match the initial datasets used in model building.

The primary objective of this research is to introduce a novel algorithm designed for incrementally updating domain-specific vocabularies. This algorithm aims to seamlessly integrate new terms from additional datasets while efficiently removing obsolete terms, and optimizing processing time and memory requirements. Unlike conventional one-time learning methods that necessitate the entire data to be available statically before building the model, this new approach facilitates ongoing updates, ensuring the vocabulary remains current without excessive resource consumption. An efficient repository will be maintained to store vocabulary terms and compact representations of previously seen labeled training sets to streamline the vocabulary update process by eliminating the need to store entire datasets and re-train models from scratch. Furthermore, the research aims to establish algorithms for assessing the incremental model's consistent performance over time, ensuring it can adapt and learn from new data without compromising previously acquired knowledge. This automated evaluation process addresses a significant research gap, providing a robust framework for assessing incremental vocabulary learning methods.

The following are the primary contributions of this work:

(1) A novel algorithm for incrementally updating a domain-specific vocabulary that can incorporate new terms from add-on datasets and remove obsolete terms. The proposed technique outperformed the conventional one-time learning method by keeping the processing time and memory requirements bounded. The proposed approach demonstrated a better F1 score than the state-of-the-art approach using a benchmark dataset.
(2) Introduction of DocLib, an efficiently designed archive for storing vocabulary terms, and a compact footprint of previously seen labeled training sets. This eliminates the need to store previously observed datasets in their entirety and train the model from scratch.
(3) Introduction of algorithms to assess the stability and plasticity of the proposed incremental learning-based approach. This study bridges a notable research gap by automating incremental updates to domain-specific vocabularies and demonstrating the stability and plasticity of incremental learning models.
(4) The generalization ability of the obtained vocabulary is illustrated by using the vocabulary obtained from one dataset to identify domain-related documents in an alternate dataset.

The effectiveness of the updated vocabulary is assessed using a task-based evaluation technique. Vocabulary terms are used to perform a downstream task of text classification. The accuracy of this task is used as a metric to measure the effectiveness of the vocabulary. A comparison of the proposed technique with state-of-the-art methods on a benchmark dataset demonstrates a better F1 score, with significantly lower memory requirements. Since a continually maintained domain-specific vocabulary can precisely capture the specialized terminology of a specific subject area or domain, it can be pivotal for a variety of tasks and applications in diverse research domains. This emphasizes the significance of this study.

The remainder of this paper is organized as follows. "Related work" section describes the related work in this area. "Methodology" section details the methodology of the proposed algorithm. "Experiments and results discussion" section discusses the experimental setup and the obtained results. "Conclusion and future work" section concludes the paper and discusses the scope of future work.

Related work

A domain model extracts relevant and useful knowledge from a related set of information sources[1]. Different terminologies have been used to refer to domain models, such as ontology[20,21], Dictionary, Vocabulary, Lexicon, and Conceptual Graph[22]. Domain models are used in various research fields, such as Artificial Intelligence, Natural Language Processing, and Information Retrieval[23,24]. Many applications deploy domain models to efficiently store and use domain knowledge. Examples include text generation[25], Risk Management[26], financial distress prediction[27], and building information modeling[28].

Incremental learning is the process of accommodating new knowledge in an existing data model without having to retrain the model from scratch[11]. Incremental learning is applied to a variety of applications in which either new data continue to be added or the available data need to be divided into small chunks. Reference[29] proposed an incremental bagging technique to detect a crypto-ransomware attack in the initial phase before

the target files are encrypted. Because the information available in the early stages of an attack is incomplete and continues to add to the progression of crypto-ransomware behavior, an incremental learning technique is employed. Reference[30] proposed an incremental summary generation algorithm that swaps sentences from the current summary and a new document to generate an improved summary. Reference[31] proposed a Chunk Incremental Cost-Sensitive Hinge Loss SVM model that distinguished the cost of false positives and false negatives in the loss function. The proposed algorithm updates an already trained model incrementally upon receiving the chunks of new data samples. Reference[32] critically reviews the current state of incremental learning in large language models (LLMs). It examines the challenges and methodologies associated with integrating new knowledge without forgetting previously acquired information. The review highlights the importance of balancing plasticity and stability to prevent catastrophic forgetting. The paper suggests further exploration into hybrid models that combine architectural and regularization strategies to improve incremental learning capabilities in LLMs. Reference[33] provides a comprehensive overview of recent advances in deep class-incremental learning, where models learn new classes without forgetting old ones. The paper identifies the primary challenge of catastrophic forgetting and suggests future research directions such as developing memory-efficient strategies. Reference[34] reviews various incremental learning methods, comparing architectural, regularization, and rehearsal strategies. The authors discuss the strengths and weaknesses of each method, emphasizing the need for more robust models capable of handling non-stationary data distributions. Reference[35] highlights the need for specialized LLMs to overcome the heterogeneity of domain data and suggests future research into more sophisticated domain adaptation techniques. Despite progress, domain specialization for LLMs still lacks standardized evaluation frameworks.

Incremental learning can be applied to update existing data models to accommodate new data. This method is more efficient than creating new models from scratch. Reference[36] partitioned streaming social network data into old and updated training tasks and incrementally learned the representation of dynamic social network data. They used GloVe to generate term vectors by building a term-context co-occurrence matrix. Any change in the frequency of the terms also requires a change in the co-occurrence matrix. The co-occurrence matrix is updated by retaining the dimensions of the matrix and performing incremental iterations on the values added between the new and old matrices. Reference[10] proposed an incremental feature selection method that updated an optimal feature subset from added-in data without forgetting previously learned knowledge. Reference[37] proposed an incremental text classification model for scenarios in which the data are not static and are possibly updated and added. Owing to the changing data, conventional methods need to retrain the model, which is a lengthy process. The proposed model required approximately 80% less training time than the conventional models. Over the past decade, the literature has extensively explored the classification of data streams using incremental learning strategies. Recent studies have highlighted persistent challenges, such as timeliness, computational complexity, and concept drift, in stream classification. To address these issues, Ref.[38] introduced a cuckoo search-based incremental binary classifier (CS-IBC). It defines class labels, accelerates class searches, and periodically updates classifiers. Testing on KDDCUP data shows CS-IBC's robustness and scalability of CS-IBC, demonstrating its effectiveness in classifying data streams. Reference[39] proposed a multidomain adaptation model based on incremental learning to transfer knowledge from the source domain to multiple target domains. Reference[40] presented HILATSA, a hybrid Arabic Twitter sentiment analysis system that adapts the lexicon to current language changes by employing both lexicon-based and machine-learning approaches to identify tweet sentiment polarities. Reference[41] addresses the universal challenge of creating adaptive models and introduces a novel incremental learning framework for an IoT-based smart water quality classification system. The model achieved state-of-the-art accuracy and low validation loss by utilizing a modified deep-learning neural network with hyperparameter tuning.

The reviewed papers reveal the following research gaps that the proposed objective can address:

(1) *Balancing plasticity and stability:* Incremental learning requires methods to balance acquiring new knowledge with retaining existing knowledge, preventing catastrophic forgetting.
(2) *Memory efficiency:* Current methods often overlook the importance of memory efficiency, which is critical for real-world applications.
(3) *Domain adaptation:* Applications (for example, LLMs) utilizing incremental learning need more effective domain adaptation techniques to handle specific domains' diverse and sophisticated requirements.
(4) *Automated evaluation:* There is a lack of robust frameworks for automated evaluation of incremental learning models, particularly in terms of their long-term performance and adaptability.

The proposed approach aims to facilitate ongoing updates while ensuring that the vocabulary remains current without excessive resource consumption. This approach inherently balances plasticity (learning new terms) and stability (retaining and efficiently managing old terms) by continuously updating the model incrementally rather than relying on one-time learning methods. This directly addresses the need for effective domain adaptation as it allows the model to integrate new terms from additional datasets seamlessly, ensuring that the vocabulary and model remain relevant and specialized to specific domains. The repository to store vocabulary terms and compact representations of previously seen labeled training sets reduces the need to store entire datasets and re-train models from scratch, optimizing memory usage and processing time. The research aims to establish algorithms for assessing the incremental model's consistent performance over time. This automated evaluation process ensures the model can adapt and learn from new data without compromising previously acquired knowledge, providing a robust framework for assessing incremental vocabulary learning methods.

Methodology

This paper proposes an incremental learning technique to update a given domain-specific vocabulary. For the experiments, the vocabulary is first created using an initial dataset. The created vocabulary is then incrementally updated using an add-on dataset. The high-level steps are illustrated in Fig. 1. The labeled dataset (both initial and add-on) may contain documents from multiple categories. These categories are treated as different domains. Step 1 in the figure shows that the initial dataset is split into training and test sets at an 80:20 ratio, and the training set is used to create an initial domain-specific vocabulary (Vocab) from scratch. The created vocabulary (Vocab) is stored in the DocLib archive. Using Vocab, the relevant information is extracted from the training set and stored as a category-specific Repository in the DocLib archive (details are provided in "Storing information in the DocLib archive" section). The vocabulary must be updated when a new dataset (referred to as an add-on dataset) related to a domain is encountered. This update may add new terms and/or delete obsolete terms. These steps are depicted in Step 2 of the figure (details are provided in "Incremental update to Vocab and Repository" section). Algorithm 1 concisely describes these steps.

"Vocabulary creation" and "Structure of DocLib archive and its usage" sections detail the precursor steps for the proposed methodology: vocabulary creation, the DocLib archive structure, and its usage. "Incremental update to Vocab and Repository" section presents the proposed incremental learning technique for updating the created vocabulary and repository. "Evaluating incrementally updated vocabulary" section describes various aspects of evaluating incremental vocabulary.

Input: D_1 (Initial Dataset), D_2 (add-on Dataset)
[Each of these datasets is split into *TrainSet & TestSet (80:20 ratio)*
D_1 *and* D_2 *TrainSet* are used for creating and updating the vocabulary, respectively.
D_1 *and* D_2 *TestSet* are used for evaluating the vocabulary*]*
Output: *DL* (DocLib archive containing *Vocab* and *Repository*)
[where *Vocab* is the incrementally updated vocabulary.
Repository - stores footprint of already seen *TrainSet*s in a compact form]

Steps:
1. *Vocab =* ***Create Vocabulary***[1] *from TrainSet of* D_1
2. *Repository = Using Vocab,* ***extract footprint of relevant information***[2] ***from TrainSet*** D_1
3. ***DL***[3] *=* ***Store Vocab and Repository in DocLib Archive***[2]
4. *To* ***incrementally update the Vocab***[4] *with Trainset* D_2*, following steps are carried out:*
   a. ***Generate Derived TrainSet from the footprint stored in Repository***[5] *and append TrainSet* D_2 *to create an Augmented TrainSet*
   b. *Vocab = Create vocabulary from the Augmented TrainSet and replace it in DL*
   c. *DL = Using Vocab, extract relevant information from Augmented TrainSet and archive the newly extracted information in DL (merging it with previous contents of Repository).*
5. *return DL*

[1] Refer to Section Vocabulary creation.
[2] Refer to Section Storing information in the DocLib archive.
[3] Refer to Section Structure of DocLib archive.
[4] Refer to Section Incremental update to *Vocab* and *Repository*.
[5] Refer to Section Retrieving information from the DocLib archive.

Algorithm 1. High-level steps for creating a vocabulary and incrementally updating it.

Vocabulary creation

The vocabulary is created by extracting selected unigrams and bigrams from a labeled domain-specific text dataset[2]. Vocabulary creation involves the following steps.

Pre-processing

The raw data in the training set are refined using the following preprocessing steps in the given sequence: Lowercase the contents; Remove accented diacritic characters, non-ASCII characters, and punctuations such as '¡,' '∈,' and '!', respectively; Apply Lemmatization to convert terms to their root form such as 'created' is converted to 'create;' and Apply Named Entity Recognition to remove names of persons to avoid mapping names of people to a domain.

Extract terms and their frequency

Part-of-Speech (PoS) tagging is used to extract <noun>-tagged unigrams and <noun, noun>-tagged bigrams as terms from a document. The noun tag is used because it provides a better representative term than other tags, such as adjectives and adverbs. Trigrams or other higher-order n-grams are not used because adding trigrams increases the vocabulary size manifold but does not significantly improve vocabulary quality for tasks such as classification[42]. For every document, the extracted terms and their occurrence frequencies in the document are recorded.

Apply frequency threshold

The frequency threshold *(u, b)* is used to filter unigrams and bigrams that occur at least u and b times across all the documents of a labeled category. $u = 3$ and $b = 3$ are used in this study according to the results obtained in Ref.[2].

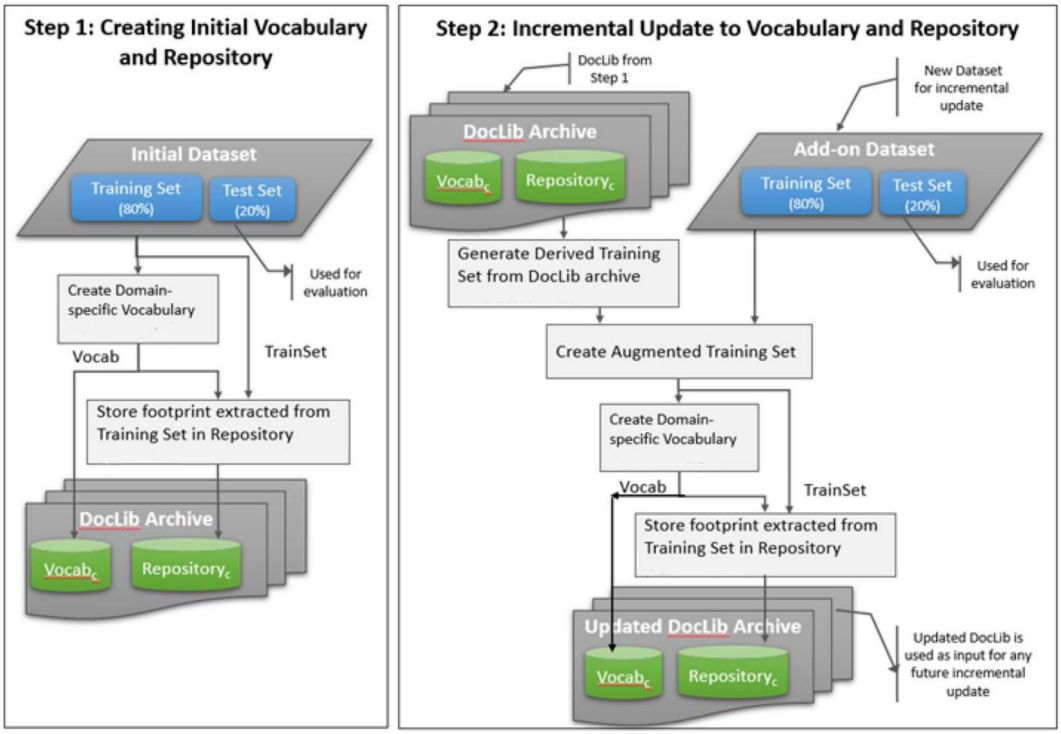

Fig. 1. Proposed methodology to create and incrementally update a domain-specific vocabulary.

Apply stochastic gradient descent (SGD) with L2 regularization

The input feature set for vectorization is formed by combining the filtered terms from all the documents in the training set. Term Frequency times Inverse Document Frequency (Tf-IDF) is then utilized to vectorize the training set, using these features as input. Tf-IDF assigns higher importance to terms that are frequent in a specific document but rare in the entire corpus. This helps highlight unique terms crucial for distinguishing domain-specific documents. In contrast, semantic-based techniques like word embeddings create dense vector representations based on word context and co-occurrence statistics and excel at capturing semantic similarities between words. Additionally, pre-trained embeddings might not precisely capture domain-specific relevance. Tf-IDF with its interpretable weights, remains valuable for understanding term importance within both the document and the entire corpus. After vectorization, the SGD model[43] with L2 regularization is trained on the obtained feature vectors using multiclass hinge loss or max-margin loss function (Eq. 1). This is equivalent to training a linear-SVM model. Ridge regularization (L2) adds the squared Euclidean norm of the model parameters (Eq. 2) as a penalty term to the loss function. This approach shrinks the model parameters toward the zero vector, thereby providing a sparse model. All features that are allocated a nonzero parameter value (coefficient) by the SGD model for a particular category c constitute the vocabulary $Vocab_c$ of that category. *Vocab* (domain-specific vocabulary) contains $Vocab_c$ (vocabulary for a particular category c) for all categories c in the labeled training set. This section uses word features and terms interchangeably.

$$Hinge\ Loss = \max\left(0, 1 - y\_i \times f(x\_i)\right), \tag{1}$$

where y_i is the true label of the data point x_i, $f(x\_i)$—predicted output for data point x_i, *max (a, b)*—maximum of *a* and *b*.

$$L2\ Regularization = \alpha 1/2 \sum_{i=1}^{m} w_i^2, \tag{2}$$

where α is a regularization parameter that controls the strength of the regularization, m—total number of weights in the model, w_i—individual weight parameters of the model.

Structure of DocLib archive and its usage

Structure of DocLib archive

The structure of the DocLib archive is illustrated in Fig. 2. For each category c of the labeled training set, the DocLib archive stores the following:

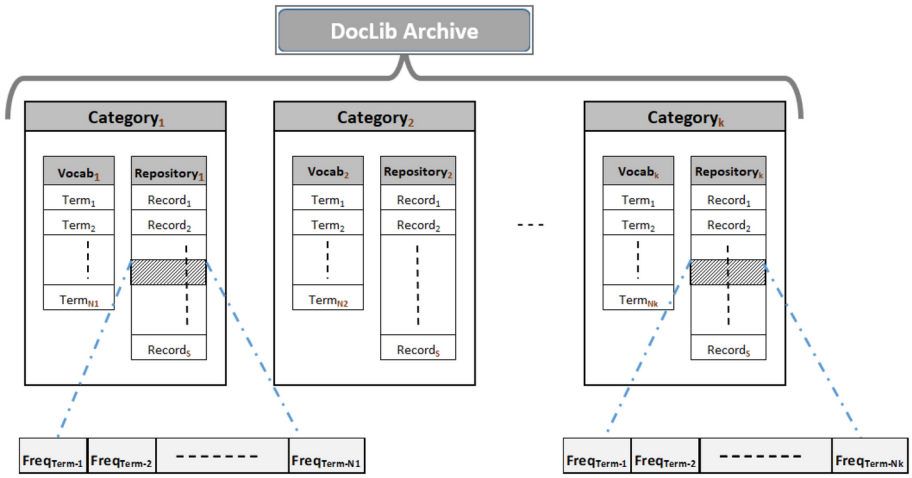

Fig. 2. Structure of the DocLib archive.

- *Vocab*$_c$: vocabulary for each category *c* (referred to as category-specific vocabulary) containing terms *Term*$_i$, where $i = 1, 2, \ldots, N_c$ and N_c denotes the total number of terms in *Vocab*$_c$.
- *Repository*$_c$: For each category *c*, there exists a dedicated repository containing a set number of records *S*. Each record holds relevant information extracted from one or more documents within the same category. This information represents the consolidated frequency of each term in the vocabulary *Vocab*$_c$ across multiple documents within that category. Initially, the frequency of each term in *Vocab*$_c$ is set to zero. "Storing information in the DocLib archive" section provides further details of computing these consolidated frequencies and storing them in a record.

The vocabulary and repository of all categories stored in the DocLib archive are collectively referred to as *Vocab* (Eq. 3) and *Repository* (Eq. 4), respectively.

$$Vocab = \bigcup_{c=1}^{k} Vocab_C, \tag{3}$$

$$Repository = \bigcup_{c=1}^{k} Repository_C, \tag{4}$$

where *k*—number of predefined categories in the training set.

The maximum memory requirement for storing the entire DocLib archive can be estimated using Eqs. (5), (6), (7), and (8).

$$\text{Memory required to store } vocab_C = MaxTermLen \times N_C, \tag{5}$$

$$\text{Memory required to store } S \text{ records in Repository}_c = S \times FreqSize \times N_C, \tag{6}$$

$$\text{Memory required per category } c = (\text{eq.5}) + (\text{eq.6}) = (MaxTermLen + S \times FreqSize) \times N_C, \tag{7}$$

$$\text{Total Memory for } k \text{ categories} = (MaxTermLen + S \times FreqSize) \times \sum_{c=1}^{k} (N_c), \tag{8}$$

where N_c—number of terms in *Vocab*$_c$, *MaxTermLen*—maximum possible length of a term in *Vocab*, *FreqSize*—number of bytes used to store the frequency of a term, *S*—number of records in a repository of the DocLib archive.

Storing information in the DocLib archive

Storing domain-specific vocabulary in *Vocab* The domain-specific vocabulary *Vocab* (containing category-specific vocabularies *Vocab*$_c$ for all *k* categories in the dataset) is stored in the DocLib archive (where each *Vocab*$_c$ stores the representative terms selected for category *c*).

When the vocabulary for each category *c* (*Vocab*$_c$) is updated incrementally, the updated *Vocab*$_c$ replaces the already stored *Vocab*$_c$ in the DocLib archive.

Storing footprint extracted from training set in repository To store a training set document d from category c in the record of the $Repository_c$, compute the frequency of each term $t \in Vocab_c$ (referred to as $freq_{t,d}$) in document d. Document d is then stored in the $Repository_c$ record in the form of these computed frequencies.

The i^{th} document of category c is stored in the $i\%S$ record of the $Repository_c$. This implies that the first S training set documents of category c are stored in S records of $Repository_c$. Subsequently, the next S training set documents of the category are merged into these S records again, continuing until all training set documents of category c are stored in the $Repository_c$. Thus, a record in the $Repository_c$ stores the information regarding multiple documents in the corresponding category. While storing a document d of category c in record r of the $Repository_c$, add the frequency of each term $t \in Vocab_c$ in d (denoted as $freq_{t,d}$) to the existing frequency of t in record r. This process merges document d into record r, consolidating the frequencies of all terms $t \in Vocab_c$ across multiple training set documents now contained in r. This method efficiently stores all category c documents from the training set within S records of $Repository_c$.

When updating the vocabulary for each category c ($Vocab_c$) incrementally, certain terms may be added or deleted from the updated $Vocab_c$. If a term is added, each record in the $Repository_c$ is extended to include the frequency of the newly added term. If a term is deleted, its frequency is dropped from each record of the $Repository_c$.

Algorithm 2 presents high-level steps for storing information in the DocLib archive.

Input: DocLib DL, TrainSet T, Vocabulary NewVocab
Output: Updated DocLib DL

UpdateDocLibArchive (DL, T, NewVocab)
1. / Store domain-specific vocabulary NewVocab in DL*/*
for each category c in NewVocab
store NewVocab$_c$ in DL

2. / Resize Repository for each term added/deleted in NewVocab*/*
for each category c in NewVocab
for each term added in NewVocab$_c$
Extend each record of the Repository$_c$ to store the frequency for this additional term
for each term deleted from NewVocab$_c$
Drop frequency of this term from each record of the Repository$_c$

3. / Store documents in T as records in the Repository */*
for each category c in T
for i^{th} document $d \in c$ in T
/ S - number of records in each repository */*
/ Merge i^{th} document with i%S record in the repository of category c */*
for each term t in NewVocab$_c$
merge/add $freq_{t,d}$ to i%S record

4. Return DL (containing Vocab and Repository updated in the above 3 steps)

Algorithm 2. Storing information in the DocLib archive.

Retrieving information from the DocLib archive

Retrieving domain-specific vocabulary from *Vocab* This simply requires accessing the terms stored in $Vocab_c$ for each category c.

Construct *TrainSet* from *Repository* stored in the DocLib archive To update the vocabulary incrementally, a *TrainSet* with S documents per category is derived from the *Repository* (which stores the footprint of the previously seen training datasets). To construct the *TrainSet*, each record r in $Repository_c$ is converted into a document d belonging to category c within the *TrainSet*. This involves adding each term t with a nonzero frequency f in r to d, repeating each term f times in d.

Incremental update to *Vocab* and *Repository*

This subsection explains the methodology depicted in Step 2 of Fig. 1. When encountering a new dataset (referred to as an add-on dataset) related to the domain, the created domain-specific vocabulary must be updated. This update may add new terms and/or delete obsolete terms.

To perform an incremental update, an augmented *TrainSet* is created by appending the derived *TrainSet* generated from the DocLib Archive (refer to "Retrieving information from the DocLib archive" section) to the training set of the newly received add-on dataset. This augmented *TrainSet* is then used to create the updated vocabulary *Vocab* following the steps listed in "Vocabulary creation" section. This updated *Vocab* is referred to as an incrementally updated vocabulary. The updated *Vocab* replaces the existing *Vocab* in the DocLib archive, and the relevant information from the augmented *TrainSet* (extracted using the steps listed in "Storing information in the DocLib archive" section) is merged in the *Repository* in the DocLib archive for any upcoming future incremental updates. Algorithm 3 describes these steps:

Input: DocLib DL, TrainSet of addon Dataset T
Output: Incrementally updated DL

IncrementalUpdateVocab (DL, T)
1. Generate DerivedTrainSet from DL
2. Augmented_Trainset = DerivedTrainSet ∪ T
3. Vocab = CreateVocabulary (Augmented_Trainset) / Refer Section Vocabulary Creation */*
4. UpdateDocLibArchive (DL, Augmented_Trainset, Vocab)
5. Return DL

Algorithm 3. Incrementally update a vocabulary using DocLib archive and add-on dataset.

The traditional approach, known as the conventional one-time model, involves retraining the model from scratch. This process utilizes the combined training sets of both the initial and add-on datasets. As more add-on datasets are introduced, the combined training set grows continuously, leading to escalating memory and processing demands without limits.

In contrast, the proposed incremental method stores the footprints of all previously observed datasets in the *Repository* of the DocLib archive using a set number of records (*S*) for each category. Whenever a new add-on dataset is received, the proposed technique generates a derived *TrainSet* with only *S* documents per category from the DocLib archive and appends it to the add-on training set to obtain an updated vocabulary. This specific step of representing the already observed *TrainSets* in terms of a fixed number of *S* documents per category (in contrast to the unbounded training-set size in the conventional approach) limits the processing and memory requirements of the proposed technique.

The overall memory requirement for the DocLib archive (refer to Eq. 8) is dependent on the number of features in the domain-specific vocabulary and not on the already seen training set sizes (as explained above). Hence, for a chosen value of *S*, *MaxTermLen,* or *FreqSize,* only N_c can vary (refer to Eq. 8) when the vocabulary is updated incrementally. The memory estimate for DocLib can change if N_c (the number of terms in the vocabulary of category *c*, $Vocab_c$) changes. However, during the incremental update, new terms may be added, and/or obsolete terms might be deleted. This keeps N_c relatively stable, ensuring that the overall memory requirements of the proposed technique are bounded.

The proposed technique is said to incrementally update the vocabulary as it archives previously learned knowledge (in the form of *Vocab* and *Repository* in DocLib) and reuses it when an add-on dataset is encountered, keeping the processing and memory requirements bounded.

The number of records *S* in each $Repository_c$ is treated as a hyperparameter. Its impact is explored in "Varying number of records (S) in a repository" section, and its optimal value is suggested. Maintaining a fixed number of records *S* for each repository helps in bounding the memory and computational requirements of the proposed algorithm. This is further discussed in "Memory requirements" and "Processing requirements" sections respectively.

Evaluating incrementally updated vocabulary

For a given labeled training set, test set, and vocabulary to be evaluated, the terms in the vocabulary are used as input to vectorize the training and test sets into feature vectors. Term frequency-inverse document frequency (Tf-IDF) is used for vectorization. The Naïve Bayes (NB) classification model is trained using the training set feature vectors. This model is used to classify the test set into predefined categories. The classification accuracy obtained on the test set is used as a metric to establish the vocabulary effectiveness. This provides insight into the ability of vocabulary terms to identify domain-related documents.

A vocabulary created on an initial dataset is updated incrementally upon receiving an add-on dataset. Analyzing the stability and plasticity of the proposed approach offers insight into the effectiveness of the updated vocabulary. Stability indicates how well a model preserves its prior knowledge. In the context of incremental vocabulary updates with each add-on training set, stability means that the accuracy of the updated vocabulary when tested on the original test set should not decrease. The detailed experimental setup for analyzing the stability of the proposed technique is presented in "Stability evaluation" section.

Plasticity is the ability of a model to acquire new knowledge regarding an ongoing task. This implies that when updating the vocabulary incrementally with each add-on training set, the corresponding updated vocabulary performs better than the previous vocabulary on the collated set of the initial and add-on test sets. The detailed experimental setup for analyzing the plasticity of the proposed technique is presented in "Plasticity evaluation" section.

Experiments and results discussion

This section is divided into the following subsections. "Experimented datasets" section describes the datasets used to create and update the vocabulary incrementally. "Varying number of records (S) in a repository" section discusses the impact of varying the number of records in the repository of the DocLib archive on vocabulary effectiveness. "Impact of performing multiple incremental updates on effectiveness of vocabulary" section studies the impact of multiple increments on the effectiveness of vocabulary. "Memory requirements" and "Processing requirements" sections present the memory and processing requirements of the proposed technique, respectively. "Stability evaluation" and "Plasticity evaluation" sections describe the stability and plasticity of the proposed

Dataset Name	Original Datasets		Experimented Datasets	
	BBC	AG	BBC (Dataset I)	AG (Dataset II)
No. of Articles	2225	127599	1,422	95,699
No. of Words	868601	3986179	548098	2977391
No. of unique Words	25136	55969	18824	48609
Average Article Size	390.4	31.2	385.4	31.1
Labelled Categories	5	4	3	3
	Business Entertainment Politics Sport Technology	World Business Sport Technology	Business Sport Technology	Business Sport Technology

Fig. 3. Dataset properties.

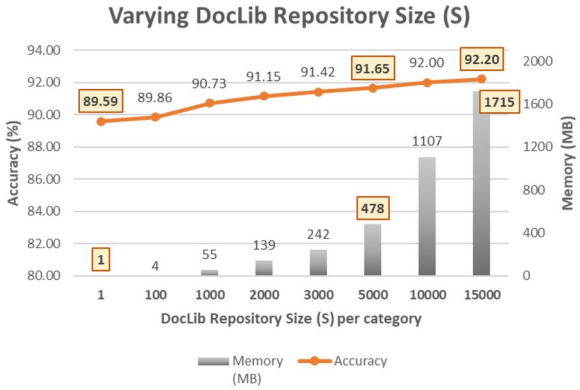

Fig. 4. The varying number of records (S) in a Repository of DocLib archive (for n = 9): Impact on accuracy and memory requirement.

technique, respectively. "Generalization of the conventional and incremental vocabulary" section discusses the generalization of the proposed technique. "Comparison with the state-of-the-art" section compares the results with those of state-of-the-art methods.

Experimented datasets

This study used two labeled benchmark text datasets: the BBC news dataset and AG's corpus of news articles. Figure 3 shows the properties of the dataset obtained from Ref.[2].

The BBC news dataset contains news articles from the BBC news website corresponding to five topics (business, entertainment, politics, sports, and technology) published from 2004 to 2005 (downloaded from http://mlg.ucd.ie/datasets/bbc.html). It is a small dataset of approximately 2000 articles with an average article size of approximately 400 words.

AG's corpus contains news articles of four categories (World, Sports, Business, and Technology) gathered from more than 2000 news sources in more than one year of activity (downloaded from http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html). It is a large dataset compared to the BBC dataset and contains approximately 1.2 lakh articles. The average article size is approximately 30 words.

News articles belonging to the categories common to both datasets are used in the experiments. Articles corresponding to the three categories of Sports, Business, and Technology are used in the experiments.

AG's corpus, which is a large dataset (~95 K articles in the experimental categories), is divided into equal chunks (the class balance for each chunk was maintained) to simulate multiple datasets for the creation and incremental update of the vocabulary. Each chunk is divided into training and test datasets at an 80:20 ratio. The first chunk is used as the initial dataset to create a vocabulary, and the remaining chunks are used as *n* add-on datasets to update the vocabulary incrementally. This is performed to simulate the real-time scenario of receiving multiple new datasets in a period and keeping the vocabulary updated with the new terminology. The BBC dataset is used to demonstrate the generalization accuracy of the incrementally updated vocabulary obtained from the AG dataset, as explained in "Generalization of the conventional and incremental vocabulary" section.

Varying number of records (S) in a repository

The DocLib archive maintains a separate repository (with *S* records) for each category to store the relevant information from the labeled training set. This number *S* is an important hyperparameter because its choice

directly influences the accuracy of the vocabulary and memory requirements. Different experiments are performed by splitting the AG dataset into initial and nine add-on datasets, with *S varying* from 1 to 15,000. The accuracy of the final vocabulary obtained for each value of *S* after nine incremental updates is evaluated. For evaluation, the training set is derived from the stored *Repository* (obtained after nine incremental updates), and the test set is the collated set of the initial and nine add-on test datasets. Algorithm 4 shows the sequence of the steps performed for a particular value of *n* and *S*. This sequence is repeated for different values of S.

Steps:

1. *{InitialDataset, D_1, D_2, …, D_n} = Divide AG dataset into an initial dataset and n add-on datasets*
2. *Vocab = Create Vocabulary from TrainSet of Initial Dataset*
3. *DL = {} /* DL: DocLib archive with a separate repository for each category */*
4. *for i = 1 to n*
   DL = IncrementalUpdateVocab (DL, TrainSet of D_i)
5. *Collated_Test = {TestSet of initialDataset}*
6. *for i = 1 to n*
   Collated_Test = Collated_Test ∪ TestSet of D_i
7. *Acc = Evaluate Vocab by using DL as TrainSet and Collated_Test as TestSet (Refer Section Evaluating incrementally updated vocabulary)*

Algorithm 4. Steps for splitting the AG dataset into initial and add-on datasets and using them to update the vocabulary incrementally.

Figure 4 presents a graph depicting the accuracy values of the final vocabularies for different values of *S* along with the memory requirements for storing the DocLib archive for each of these experiments. The conventional one-time model achieved an accuracy of 92.31%, with a memory requirement of 28 MB. The following observations can be made from Fig. 4.

- *Accuracy—increases with the increasing number of records (S) in the Repository of the DocLib archive*. With increased S, the accuracy of the final vocabulary built after $n=9$ increments increases, but with an increased memory requirement. The accuracy increased with increasing *S* because a higher value of S results in a smaller number of documents merged per record in the repository, thereby making it a comparatively closer approximation of the actual training sets used to create and update the vocabulary incrementally.
- *Optimal number of records in the Repository of the DocLib archive:* $S=5000$ yielded a good accuracy of 91.65% (0.66% less than that of the conventional one-time model) for a bounded memory requirement of 478 MB. Smaller values of $S=1$ and 100 did not provide encouraging accuracy values because of the high compaction factor in the DocLib archive. Larger values of $S=10{,}000$ and 15,000 have a slightly higher accuracy but require significantly more memory. Therefore, $n=10$ and $S=5000$ are used for further experiments.

Impact of performing multiple incremental updates on effectiveness of vocabulary

To evaluate the effectiveness of a vocabulary updated incrementally with new add-on datasets, an experiment is conducted by splitting the AG dataset into an initial dataset and $n=9$ add-on datasets. The number of records (*S*) in a *Repository* is 5000 (the choice of *S* is explained in "Varying number of records (S) in a repository" section). Algorithm 5 (Steps 1–5) evaluates the effectiveness of each vocabulary obtained after performing individual incremental updates. Figure 5 presents a graph depicting the accuracy of these vocabularies ($BaselineAcc_i$ of Algorithm 5). The effectiveness of the vocabularies obtained after performing incremental updates increased during the first few increments and was then maintained in subsequent increments. The final incrementally updated vocabulary obtained after incorporating all add-on datasets has an accuracy of 91.65%, which is marginally lower (0.66%) than the accuracy of the vocabulary created using the conventional one-time approach

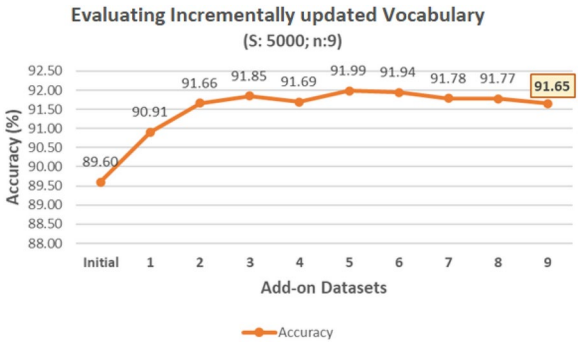

Fig. 5. Evaluating incrementally updated vocabularies with 9 add-on datasets.

(92.31%). However, vocabulary updated incrementally has the benefit of bounded memory and processing requirements in comparison to the vocabulary created using a conventional one-time approach (see "Memory requirements" and "Processing requirements" sections).

Memory requirements

For each category, the corresponding repository of the DocLib archive has a fixed number of records *S*. The memory requirements for storing the DocLib archive for different values of *S* are shown in Fig. 4 in "Varying number of records (S) in a repository" section. The salient points related to the memory requirements of the proposed technique are as follows:

- *Benefit of bounded memory requirement:* The memory requirement for the DocLib archive increases with an increase in *S*. However, for a given value of *S*, the memory requirement for the DocLib archive remains bounded. This is because relevant information from the training set of any future add-on dataset is merged into *S* records only. The DocLib archive is designed in such a way (refer to "Structure of DocLib archive" section) that, irrespective of the number of add-on datasets merged, its memory requirements remain relatively bounded for a given number of records (*S*) in a repository. Only a slight change in the overall memory requirement can occur because of the additional terms encountered in the add-on dataset and/or the removal of obsolete terms from the vocabulary.
- *Comparison with the memory requirement of the conventional one-time model:* The conventional one-time model must store all the training sets of previously used datasets for any future update to the vocabulary. The vocabulary is then recreated from scratch using all the training sets as a single dataset. As new add-on datasets are encountered, the training set size continues to grow, and accordingly, the memory requirements grow unboundedly.

Processing requirements

The vocabulary creation steps, as explained in "Vocabulary creation" section, involve preprocessing the dataset, extracting features and their frequencies, applying a frequency threshold, and finally, performing feature selection using SGD with L2 Regularization. The processing time for these steps is directly proportional to the number of documents in the training set used to create the vocabulary.

When the new add-on datasets are received, using the conventional one-time method, the size of the training set continues to increase. Hence, the processing time for vocabulary creation continues to increase.

The proposed technique involves combining the training set derived from the DocLib archive, which contains a fixed number of records (*S*) for each category, with the training set of the add-on dataset. This ensures that the total number of documents in the combined training set is bounded. Hence, the processing time for vocabulary creation using the proposed incremental technique (for any fixed value of S) for multiple add-on training sets remained bounded.

Stability evaluation

Stability is the ability of a model to conserve the previously learned knowledge of old tasks. This means that by incrementally updating the vocabulary with the add-on datasets, the accuracy of the updated vocabulary on the previous test datasets should not be adversely impacted. Different experiments are performed by splitting the AG dataset into initial and $n = 9$ add-on datasets with $S = 5000$.

Figure 6 shows the baseline and stability curves. The computational details of these curves are presented in Algorithm 5. It can be observed from the figure that the accuracy values on the stability curve are quite close to the baseline curve in Fig. 6, with a maximum decrease of just 0.72%, across all incremental updates. Hence, the vocabulary has been learned from the add-on datasets without losing much of its effectiveness on the previous datasets. This indicates that the proposed approach is stable.

Algorithm 5 details the sequence of the steps performed to evaluate the stability of the proposed technique.

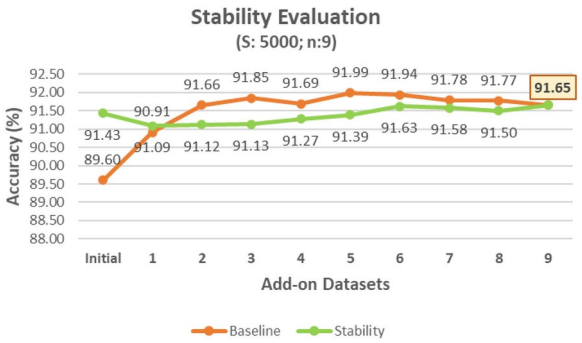

Fig. 6. Evaluating stability as the vocabulary is updated incrementally with add-on datasets.

Steps:

1. *{InitialDataset, D_1, D_2, …, D_n} = Divide AG dataset into initial dataset and nine add-on datasets*
2. *Vocab = Create Vocabulary from TrainSet of Initial Dataset*

3. *DL = {} /* DL – DocLib archive */*
4. *$Collated\_Test_0$ = {TestSet of InitialDataset}*

5. *for i = 1 to n*
   DL = IncrementalUpdateVocab (DL, TrainSet of D_i)
   $Collated\_Test_i$ = $Collated\_Test_{i-1}$ ∪ TestSet of D_i
   */*Baseline curve is plotted using $BaselineAcc_i$. For details of the evaluation, refer to Section Evaluating incrementally updated vocabulary */*
   $BaselineAcc_i$ = Evaluate Vocab by using Derived TrainSet generated from DL as TrainSet and $Collated\_Test_i$ as TestSet
   / DL obtained after n iterations of above for loop is used as TrainSet for evaluating stability*/*

   */*Stability Evaluation*/*
6. *for i = 1 to n*
   */*The Stability curve was plotted using $StabilityAcc_i$. For details of the evaluation, refer to Section Evaluating incrementally updated vocabulary */*
   $StabilityAcc_i$ = Evaluate Vocab by using Derived TrainSet generated from final DL as TrainSet and $Collated\_Test_i$ as TestSet

Algorithm 5. Steps for evaluating the stability of the proposed incremental learning technique.

Plasticity evaluation

Plasticity is the ability of a model to acquire new knowledge about an ongoing task. This implies that when updating the vocabulary incrementally with each add-on training set, the corresponding updated vocabulary performs better than the previous vocabulary on the collated test set of the initial and add-on test sets. Different experiments are performed by splitting the AG dataset into initial and nine add-on datasets, with $S = 5000$.

Algorithm 6 details the sequence of the steps performed to evaluate the plasticity of the proposed technique.

Steps:

1. *{InitialDataset, D_1, D_2, …, D_n} = Divide AG dataset into initial dataset and nine add-on datasets*
2. *Vocab = Create Vocabulary from TrainSet of Initial Dataset*

3. *DL = {} /* DL – DocLib archive*/*
4. *for i = 1 to n*
   Collated_Test = Collated_Test ∪ TestSet of D_i

   */*Plasticity Evaluation*/*
5. *for i = 1 to n*
   DL = IncrementalUpdateVocab (DL, TrainSet of D_i)
   / DL obtained in the i^{th} iteration of this for loop is used to generate derived TrainSet for evaluating Plasticity*/*
   */*Plasticity curve is plotted using $PlasticityAcc_i$. For details of the evaluation, refer to Section Evaluating incrementally updated vocabulary */*
   $PlasticityAcc_i$ = Evaluate Vocab by using DL to generate derived TrainSet and Collated_Test as TestSet

Algorithm 6. Steps for evaluating the plasticity of the proposed incremental learning technique.

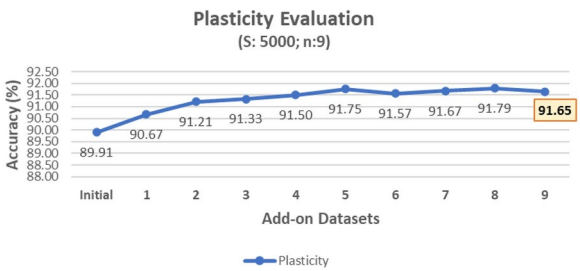

Fig. 7. Evaluating plasticity as the vocabulary is updated incrementally with add-on datasets.

Memory range	Learn# model		Classification model (vocabulary from proposed technique)	
	Learn# technique (varying student model)	F1 score	Proposed technique	F1 score
1 to 50 MB	S (TF-IDF + XGBoost) + R + T − XGBoost (Memory 30 MB)	0.7741	100 records (Memory 4 MB)	0.89823
50–500 MB	S (Word2Vec + LSTM) + R + T (Memory 500 MB)	0.8381	5000 records (Memory 478 MB)	0.91621
500–4200 MB	S(BERT) + R + T (Memory 4150 MB)	0.9436	15,000 records (Memory 1660 MB)	0.92189

Table 1. Comparison of the F1 scores of the models (Learn# model vs proposed technique) with similar memory requirements.

Figure 7 plots the plasticity accuracy values described in Algorithm 6. The accuracy of the incrementally updated vocabularies obtained after learning from each add-on dataset improves consistently. Hence, the vocabulary has been learned from each add-on dataset, which establishes the plasticity of the proposed approach.

Generalization of the conventional and incremental vocabulary

To showcase the generalization ability of the incremental vocabulary generated through the proposed method, the AG dataset is divided into an initial dataset and nine add-on datasets. Initially, a vocabulary is created using the initial dataset and subsequently updated incrementally with the nine add-on datasets. After nine iterations, the resulting vocabulary is then evaluated using the training and test sets of the BBC dataset (see "Evaluating incrementally updated vocabulary" section for evaluation specifications).

This experiment yielded an accuracy rate of 97.89%, comparable to the accuracy achieved by creating a vocabulary solely from the BBC dataset's training set and evaluating it on the BBC test set. This suggests that the obtained incremental vocabulary effectively captures key terms from the domain, enabling it to discern domain-related documents from diverse sources.

Comparison with the state-of-the-art

In the proposed technique for incrementally updating a domain-specific vocabulary, the AG corpus is divided into an initial dataset and nine add-on datasets. The updated vocabulary obtained after nine increments is evaluated by building a Naïve Bayes (NB) classification model using the steps described in "Evaluating incrementally updated vocabulary" section. The training set used to build the NB model is derived from the DocLib archive, which is obtained after nine incremental updates. The test set is a collated set of nine add-on test datasets. Different experiments are performed by varying the number of records in the repository of the DocLib archive to $S = 100$, 5000, and 15,000. Table 1 presents the F1 scores of the Naïve Bayes classification models and the storage requirements of the corresponding DocLib archive in different experiments.

These NB models are compared to Learn# models in terms of F1 score, with the Learn# models requiring a similar range of memory as the corresponding DocLib archive needed to create the NB models. Learn# has the following four components. First, the Student model (S) extracts text features and makes predictions. Second, a Reinforcement Learning module (R) filtered the predictions. Third, the Teacher model (T) reclassified the results of the student models to improve classification. Finally, the discriminator model (D) eliminates similar student models. Learn# outperforms many conventional one-time methods on the AG corpus. The effectiveness of Learn# for classification is validated by experimenting with different models, such as S (TF-IDF + XGBoost) + R + T − XGBoost classifier based on TF-IDF features used as the student model (S) for text classification, followed by R and T models, S (Word2Vec + LSTM) + R + T—100-dimensional Word2Vec vectors used with LSTM as the Student model, S (BERT) + R + T—BERT model used for paired-sentence classification as the student model, and S (BERT) fed into the R and T models to obtain the final predictions. Table 1 lists the F1 scores and memory requirements for the three models.

The classification models built using incremental vocabulary demonstrated better F1 scores with significantly lower memory requirements than the Learn# models. Learn# used a discriminator model to control the number of student models by deleting similar ones. This implies that as the number of student models increases, they might lose many previously learned models to keep the memory and processing requirements bounded. However, the proposed technique stores the relevant information from all previously observed training sets in the DocLib archive.

Reference[44] introduces a novel optimization method, MAS (mixing ADAM and SGD), which simultaneously combines the strengths of two different optimizers, ADAM, and SGD. MAS leverages both optimizers concurrently, capitalizing on their respective advantages. A neural architecture based on BERT (renowned as one of the most effective methods for processing text documents) is utilized for experiments. A pre-trained iteration of BERT is leveraged to expedite the training, allowing fewer epochs to be executed. The accuracy results of the BERT model pre-trained on AG news (10 epochs) after five runs using the SGD optimizer are reported, with a maximum accuracy of 91.39%.

The accuracy achieved by our proposed approach is 91.65% with S (number of records) = 5000, requiring a reasonable and bounded memory of 478 MB. The accuracy increased further with increasing S. This indicates that the vocabulary generated and maintained using the proposed approach can identify domain-related documents with reasonably good accuracy. In addition, integrating advanced optimizers, such as MAS, into the proposed approach can further improve the effectiveness of vocabulary and can be pursued in the future.

Reference[35] emphasizes the need for domain specialization in LLMs to handle the heterogeneity of domain data, sophisticated domain knowledge, and unique objectives. Reference[32] discusses using incremental learning

for designing LLM-based learning systems. It describes various challenges associated with incremental learning, such as maintaining high accuracy while learning from streaming data without accessing the entire dataset, balancing plasticity, and stability to prevent catastrophic forgetting, efficiently managing memory resources to handle continuous data inflow, and developing methods for sequential data processing to handle big data effectively. Our proposed method's continual incremental updates and efficient resource management offer a practical solution to the challenges discussed. The proposed algorithms for stability-plasticity analysis provide robust frameworks for monitoring and maintaining the performance of incremental learning-based models. The method demonstrates reasonably good accuracy after processing the AG dataset in chunks ("Impact of performing multiple incremental updates on effectiveness of vocabulary" section), balances stability and plasticity effectively ("Stability evaluation" and "Plasticity evaluation" sections), and generalizes well to the BBC dataset ("Generalization of the conventional and incremental vocabulary" section). These results validate the method's effectiveness in real-world incremental learning scenarios, addressing key challenges and contributing to robust and efficient LLMs.

Key findings and relevance

The following are the key findings of this work:

- The incremental vocabulary generated by the proposed method using the AG dataset demonstrated strong generalization, achieving a high accuracy rate of 97.89% on the BBC dataset, comparable to vocabulary created from the BBC training set.
- The Naïve Bayes models built with the incremental vocabulary have shown better F1 scores and significantly lower memory requirements in comparison to the state-of-the-art model like Learn#. The proposed technique efficiently stored relevant information from all previous training sets in the DocLib archive, avoiding the loss of learned models, unlike the memory management approach in Learn#.
- Using the SGD optimizer with a BERT-based architecture on the AG dataset yielded an accuracy of 91.39%. However, the accuracy achieved by our proposed approach is 91.65% with S (number of records) = 5000.
- The proposed method effectively balanced stability and plasticity, preserving previously learned knowledge while successfully acquiring new knowledge, as demonstrated by its performance on incremental updates and generalization to the BBC dataset.
- The proposed approach addressed key challenges in incremental learning, such as maintaining accuracy, managing memory and computation requirements, and handling continuous data inflow, supported by robust stability-plasticity analysis frameworks.

To generalize the proposed approach to more recent architectures like deep neural networks in the future, the proposed incremental vocabulary updates can be integrated with the embedding layers to incorporate new vocabulary rather than updating the entire model. This can enable scalable training by processing data in small batches as they arrive, reducing computational demands and making it feasible for large-scale applications. Static models may struggle with personalized or context-specific data, whereas the proposed incremental approach can be used to adapt by continuously learning from the relevant data. Using the footprint of the previously seen datasets from DocLib can reduce the training times and memory requirements of deep neural networks when receiving new data. It can prevent catastrophic forgetting by retaining old knowledge while integrating new information and handles concept drift by adapting to changes in data distribution over time.

Conclusion and future work

This study proposed a technique for incrementally updating domain-specific vocabularies. This technique applies the principles of Incremental Learning to update a given domain-specific vocabulary. It stores previously observed training sets using vocabulary terms in a compact archive named, DocLib. This paper presents an algorithm to store relevant information in the DocLib archive and then incrementally update the vocabulary using the DocLib archive and the add-on dataset. The structure of the DocLib archive is meticulously designed to keep the memory requirements and processing time bounded. The updated vocabulary is evaluated using its terms as inputs to create a text classifier. The classification accuracy is used as a metric to evaluate the effectiveness of the vocabulary. The impact of varying the number of records (S) in the repository of the DocLib archive on the accuracy and memory requirements of the incrementally updated vocabularies is evaluated, and an optimal value of $S = 5000$ records per category is proposed. Experiments are conducted to study the impact of multiple increments on vocabulary effectiveness. Incorporating incremental updates from multiple add-on datasets keeps the vocabulary updated with the latest domain terminology and maintains its effectiveness. Experiments on the benchmark AG dataset demonstrate that the proposed technique outperforms the conventional one-time method, which recreates vocabulary from scratch by keeping the processing time and memory requirements bounded. Subsequently, the stability and plasticity of the incremental vocabulary obtained using the proposed technique are validated. The generalization of the incrementally updated vocabulary obtained from the AG dataset is demonstrated using the BBC dataset. A task-based evaluation of the proposed technique and its comparison with the state-of-the-art incremental classification model (Learn#) built on AG's corpus demonstrate the effectiveness of the proposed technique.

This work can be extended in the future to include collaboration with other classification algorithms, such as decision trees and random forests. Also, the incrementally learned vocabulary can be applied to build a deep neural network-based multiclass text classification model. The model's performance can be compared when built with the incrementally obtained vocabulary versus learning the vocabulary at once using a benchmark method. This will further validate the effectiveness of the proposed technique in contrast to the conventional one-time learning methods and demonstrate its applicability. Other applications, such as sentiment analysis

and blog categorization, can be explored for task-based evaluation of the updated vocabulary. Additionally, the generalization accuracy of the vocabulary can be evaluated using additional datasets.

Data availability

All data that support the findings of this study are included (links given) within the article.

Code availability

All the designed algorithms are given in the manuscript.

Received: 25 April 2024; Accepted: 4 November 2024
Published online: 02 January 2025

References

1. Clark, M. et al. Automatically structuring domain knowledge from text: An overview of current research. *Inf. Process. Manag.* **48**, 552–568 (2012).
2. Sood, M., Gera, J. & Kaur, H. Creation, evaluation, and optimization of a domain-based dictionary. *J. Intell. Fuzzy Syst.* **43**, 6123–6136 (2022).
3. Razia Sulthana, A. & Ramasamy, S. Ontology and context based recommendation system using Neuro-Fuzzy Classification. *Comput. Electr. Eng.* **74**, 498–510 (2019).
4. Gutiérrez-Batista, K., Campaña, J. R., Vila, M. A. & Martin-Bautista, M. J. An ontology-based framework for automatic topic detection in multilingual environments. *Int. J. Intell. Syst.* **33**, 1459–1475 (2018).
5. Xing, F. Z., Pallucchini, F. & Cambria, E. Cognitive-inspired domain adaptation of sentiment lexicons. *Inf. Process. Manag.* **56**, 554–564 (2019).
6. Kalra, V., Kashyap, I. & Kaur, H. Classification based topic extraction using domain-specific vocabulary: A supervised approach. *Indones. J. Electr. Eng. Comput. Sci.* **26**, 442–449 (2022).
7. Kalra, V., Kashyap, I. & Kaur, H. Improving document classification using domain-specific vocabulary: Hybridization of deep learning approach with TFIDF. *Int. J. Inf. Technol.* **14**, 2451–2457 (2022).
8. Kim, Y. et al. Validation of deep learning natural language processing algorithm for keyword extraction from pathology reports in electronic health records. *Sci. Rep.* **10**, 20265 (2020).
9. Jantscher, M. et al. Information extraction from German radiological reports for general clinical text and language understanding. *Sci. Rep.* **13**, 2353 (2023).
10. Losing, V., Hammer, B. & Wersing, H. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing* **275**, 1261–1274 (2018).
11. Yang, Y., Chen, D., Zhang, X., Ji, Z. & Zhang, Y. Incremental feature selection by sample selection and feature-based accelerator. *Appl. Soft Comput.* **121**, 108800 (2022).
12. Agarwal, S., Rattani, A. & Chowdary, C. R. A-iLearn: An adaptive incremental learning model for spoof fingerprint detection. *Mach. Learn. Appl.* **7**, 100210 (2022).
13. Amirat, Y. et al. Assistance and service robotics in a human environment. *Robot. Auton. Syst.* **75**, 1–3 (2016).
14. Hammer, B. & Toussaint, M. Special issue on autonomous learning. *Künstliche Intell.* **29**, 323–327 (2015).
15. Menegatti, E., Berns, K., Michael, N. & Yamaguchi, H. Special issue on intelligent autonomous systems. *Robot. Auton. Syst.* **74**, 297–298 (2015).
16. Thrun, S. Toward robotic cars. *Commun. ACM* **53**, 99–106 (2010).
17. Rico-Juan, J. R. & Iñesta, J. M. Adaptive training set reduction for nearest neighbor classification. *Neurocomputing* **138**, 316–324 (2014).
18. Ditzler, G. & Polikar, R. Incremental learning of concept drift from streaming imbalanced data. *IEEE Trans. Knowl. Data Eng.* **25**, 2283–2301 (2013).
19. Elwell, R. & Polikar, R. Incremental learning of concept drift in nonstationary environments. *IEEE Trans. Neural Netw.* **22**, 1517–1531 (2011).
20. Navigli, R. & Velardi, P. *From Glossaries to Ontologies: Extracting Semantic Structure from Textual Definitions*, vol. 167, 71–87 (2008).
21. Alruqimi, M. & Aknin, N. Bridging the gap between the social and semantic web: Extracting domain-specific ontology from folksonomy. *J. King Saud Univ. Comput. Inf. Sci.* **31**, 15–21 (2019).
22. Sowa, J. F. Conceptual graphs. In *Handbook of Knowledge Representation, Foundations of Artificial Intelligence* 213–237 (2008).
23. Akbik, A. & Bross, J. B. G. *Wanderlust: Extracting Semantic Relations from Natural Language Text Using Dependency Grammar Patterns* (2009).
24. Wilks, Y. & Brewster, C. Natural language processing as a foundation of the semantic web. *Found. Trends Web Sci.* **1**, 1 (2009).
25. Fuertes-Olivera, P. & Bergenholtz, H. Dictionaries for text production. In *The Routledge Handbook of Lexicography* (ed. Fuertes Olivera, P.) 267–283 (Routledge, 2018).
26. Xu, N. et al. Relation extraction of domain knowledge entities for safety risk management in metro construction projects. *Buildings* **12**, 1 (2022).
27. Li, S., Shi, W., Wang, J. & Zhou, H. A deep learning-based approach to constructing a domain sentiment lexicon: A case study in financial distress prediction. *Inf. Process. Manag.* **58**, 102673 (2021).
28. Herrera-Martín, J. J., Castilla-Rodríguez, I., González, E. J. & Martín-Dorta, N. A method for transferring BIM data into domain ontologies: A case study based on airport services. *Egypt. Inform. J.* **23**, 447–467 (2022).
29. Al-rimy, B. A. S., Maarof, M. A. & Shaid, S. Z. M. Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. *Future Gener. Comput. Syst.* **101**, 476–491 (2019).
30. Chowdary, R. & Kumar, P. *An Incremental Summary Generation System* 83–92 (2008).
31. Gu, B., Quan, X., Gu, Y., Sheng, V. S. & Zheng, G. Chunk incremental learning for cost-sensitive hinge loss support vector machine. *Pattern Recogn.* **83**, 196–208 (2018).
32. Jovanovic, M. & Voss, P. Towards incremental learning in large language models: A critical review. Preprint at https://arxiv.org/abs/2404.18311 (2024).
33. Zhou, D.-W. et al. Deep class-incremental learning: A survey. Preprint at https://arxiv.org/abs/2302.03648 (2023).
34. Luo, Y., Yin, L., Bai, W. & Mao, K. An appraisal of incremental learning methods. *Entropy* **22**, 1 (2020).
35. Ling, C. et al. Domain specialization as the key to make large language models disruptive: A comprehensive survey. Preprint at https://arxiv.org/abs/2023.07.11 (2023).
36. Peng, H. et al. Incremental term representation learning for social network analysis. *Future Gener. Comput. Syst.* **86**, 1503–1512 (2018).
37. Shan, G., Xu, S., Yang, L., Jia, S. & Xiang, Y. Learn#: A novel incremental learning method for text classification. *Expert Syst. Appl.* **147**, 113198 (2020).

38. Abdualrhman, M. A. A. & Padma, M. C. CS-IBC: Cuckoo search based incremental binary classifier for data streams. *J. King Saud Univ. Comput. Inf. Sci.* **31**, 367–377 (2019).
39. Wei, X. et al. Incremental learning based multi-domain adaptation for object detection. *Knowl. Based Syst.* **210**, 106420 (2020).
40. Elshakankery, K. & Ahmed, M. F. HILATSA: A hybrid Incremental learning approach for Arabic tweets sentiment analysis. *Egypt. Inform. J.* **20**, 163–171 (2019).
41. Nemade, B. & Shah, D. An efficient IoT based prediction system for classification of water using novel adaptive incremental learning framework. *J. King Saud Univ. Comput. Inf. Sci.* **34**, 5121–5131 (2022).
42. Sood, M., Kaur, H. & Gera, J. Creating domain based dictionary and its evaluation using classification accuracy. In *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)* 341–347. https://doi.org/10.1109/INDIACom51348.2021.00059 (2021).
43. Zhang, T. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the Twenty-First International Conference on Machine Learning* 116. https://doi.org/10.1145/1015330.1015332 (Association for Computing Machinery, 2004).
44. Landro, N., Gallo, I. & Grassa, R. L. Mixing ADAM and SGD: A combined optimization method. https://arXiv.org/abs/2011.0 (2020).

Author contributions

M. J. wrote the main manuscript text and prepared the experiments, figures, and tables. H. K. supervised conceptualization and writing, and B. G., J. G. and V. K. contributed to the discussions and interpretation of the data. All authors reviewed the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to B.G.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.