Micro-learning based content curation using Artificial Intelligence for Learning Experience Platform

Kaklij Vaishnavi Arun
Department of Computer Science and Engineering, SOCSE, SandipUniversity, Nashik, India

Mr.Kunal V. Shah
Visiting faculty at CDAC-ACTS, Pune

Prof. Umakant Mandawkar
Assistant professor
Department of Computer Science and Engineering, SOCSE, SandipUniversity, Nashik, India

---

Abstract— The internet is a vast pool of all kinds of educational information, where every video, blog, article, podcast, and webpage is possibly found within a matter of a click. From this wide range of available content, it is difficult for the learner to search or find relevant information without spending hours and hours refining the search results. Learners want to learn something quick, something small, up-to-date, and above all, at their own pace. They want bite-sized information that is focused, easily absorbed, and available on-demand, on any device based on their interest and knowledge. This has led to the necessity of analyzing the gathered data and then curating valuable information from it. In this paper, the field of learning is examined regarding various challenges learners are facing. Additionally, a few of the challenges which are encountered in the literature will be tackled in the proposed work.

Hence the objective of this paper is to provide a platform which will help learners to find the right learning materials by suggesting them with content specifically targeted to the specific domain based on their skills, strengths, weaknesses, backgrounds, needs, different levels of domain knowledge, learning style, experience and interest in micro-form using the principles of micro-learning using AI-based Content curation.

Keywords ---Macro-learning, Micro-learning – Text summarization, AI-based Content curation - Context-based filtering, Auto-tagging, and classification.

I. INTRODUCTION

"Plenty" is a problem. Learners find it most difficult to learn, NOT because there isn't enough content, but because there is TOO MUCH of it, and they cannot find what is valuable and up-to-date. Learners - they can't get access to what they need when they need it. So, the learning experience platform needs an **intelligent method** to make a move from long, information-based online courses to **micro-learning**, which is the outcome of **content curation**. It helps **students** to find the correct and on-the-go learning materials for their current challenges fast and precise so that they can utilize their precious time on mastering them. It reaches to time-poor **employees** by delivering relevant micro-content which is short, up-to-date, and personalized so that the employee can access it when and where they need it the most.

Micro-learning-based Content curation:

Our studies indicate that in 10 minutes, the eager learners want to brush-up on their knowledge and skills. But the learners are not able to find relevant training in the maze. It happens when the average length of the course is more than 60 minutes. Imagine if the average course duration is 10 minutes when organizations adopt micro-learning then there will be at least 8 courses for every 1 course. That's why curated micro-learning content is becoming the main focus area in many L&D organizations.

The micro nature of the content ensures that it can fit into the limited time that learners have. Linking the content into pathways makes sure that even though each chunk is small, together multiple chunks can help learners meet their learning goals. With a deep-rooted habit of learners of searching content on the Internet for bite-sized learning, today's employees and students learn best not from conference room seminars or long certified courses respectively but from bite-sized information known as microlearning— which is digestible (easy to understand and remember) and that they can apply right away.

II. RELATED WORK AND TECHNOLOGIES USED

There are various researches and studies, which have been done over. Some of the studies and implementations from other researchers are described in this survey paper by Kaklij, V.A., Shah, M.K.V., and Mandawkar, M.U., 2019. "MICROLEARNING BASED CONTENT-CURATION USING ARTIFICIAL INTELLIGENCE FOR LEARNING EXPERIENCE PLATFORM: A SURVEY."
This is an overview of the challenges which are encountered in the survey. [1]

ACTORS	CHALLENGES
Employee	1. The world of work is changing rapidly, if we don't keep pace with changes in our sector, we will render ourself **outdated** over time. 2. **TIME** : "Content shock problem:Too much content , too little time."
Student	**Forgetting Curve i.e** The phenomenon of learning and promptly forgetting information in a matter of days unless reviewing it again.
Employee + Student	1. **Quantity** : "Plenty is the problem " 2. **Personalised** : Learners arrive at workplace with dramatically different backgrounds. 3. **Information search task** have different levels of difficulty to learners: Time limitation , No experience, complexity, No interest.

A few of the above-mentioned challenges will be tackled in the proposed work.

I am targeting the learners who are mostly in short of time to go and can't study the entire course, so they will get to learn the study material in micro-form. A micro-learning-based content curation approach can be the best-fit solution for all the issues mentioned in the above use-cases. Hence, this proposed model/ system will make use of this approach to design a generic model, this will have micro-learning content.

TECHNOLOGIES USED:

A. AI-based Filtering

Over the years the quantity of information available to us has grown into something quite overwhelming. Content is directly on the web which is great as it makes it very accessible but you don't need too long to understand that there is more content than there is time to read it. Sometimes, more underrated content goes unnoticed and often gets overcrowded with content in which you get bombarded regularly with unwanted content.[6] With the fast-growing quantity of information on the Internet and a considerable number of learners, companies must scan, search, filter, and provide personalized and contextualized content to the learners according to their needs and tastes.

One solution to this is to use information filtering.

Information filtering (IF) objective is to remove redundant and/or unnecessary information from data streams.[6] Years ago, information filtering was mostly adopted by governments as a means to censor information but now in the age of information, it's all about bringing the right information to the right person. Information filtering is especially applied to semi-structured / unstructured data (e.g. E-mail messages, documents), mostly textual data. They are made to be able to handle a large amount of data and are based on a user profile.

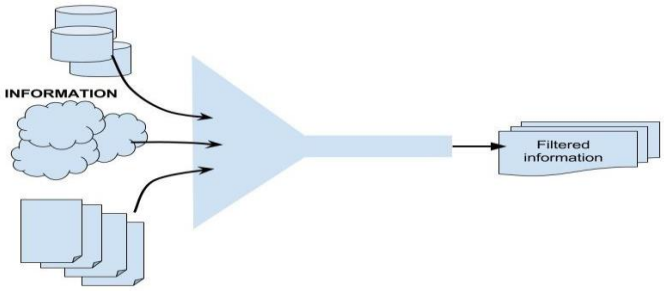


Figure 1: Information Filtering [8]

To provide learners with service or recommending learning materials, recommendation engines use algorithms. Lately, these engines have started applying machine learning algorithms making the predicting process of items more accurate. Depending on the data received from recommendation systems, the algorithms change. [5]

Machine learning algorithms for filtering are generally divided into the following categories:
Collaborative, content-based, context-based filtering.

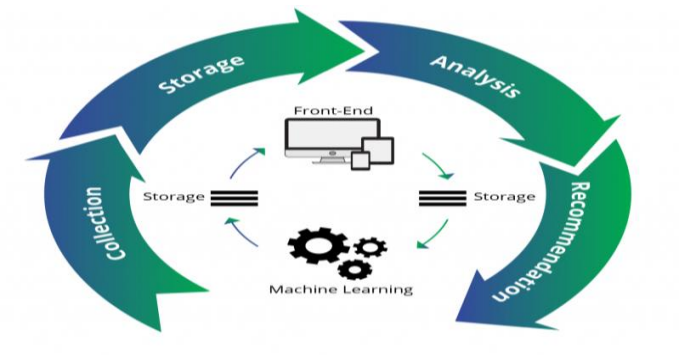


Figure 2: Process of Filtering [7]

System processes data through these four steps: collecting, storing, analyzing, and filtering. [7]

1. Collecting the data

The first step in building a recommendation engine is data gathering. Data is classified into explicit and implicit. Data given by users, like comments and ratings are explicit. While, implicit data may include return history, search log, page views, clicks, and cart events. This kind of data is collected from any users who visit the given website. Collecting behavioral data is not difficult since you can keep user activities logged on your website. As each user likes or dislikes various items, their datasets are different. After a while, the engine becomes cleverer, when it's fed with more data. And the recommendations become more suitable too, so the learners are more inclined to view, click, and read. [5]

2. Storing the data:

To have improved recommendations, you should generate additional data for the algorithms you use. You can choose what storage type you need to use with the help of the data you use for creating recommendations. It is up to you whether to use Hadoop or a NoSQL database or a standard SQL database or even some sort of object storage. All of these alternatives are practical and conditioned with whether you capture input or user behavior. A managed and scalable database reduces the number of essential tasks to minutest and targets on the recommendation itself.

3. Analyzing the data

To discover articles with the same user engagement data, it is necessary to filter it by using various analyzing techniques. Sometimes it's necessary to cater immediately when the user is reading the article, so that type of analysis is required.

Some of the ways to analyze this kind of data are as follows:

a) **Real-time system**
b) **Near-real-time analysis**
c) **Batch analysis**

4. Filtering the data

The next step is filtering the data to provide relevant recommendations to the learners. For implementing this method, you should select an algorithm appropriate for the engine you use.

Algorithms to perform filtering:

i. **Context-aware:** Big Data has limited benefits if not combined with its extra intelligent sibling, Context. When looking at unstructured data, suppose, we encounter the number "31" and have no clue what that number means, whether it is the number of dollars a stock increased over the past week, the number of days in the month, or the number of items sold today. This number "31" could convey anything, without the layers of context that describes what type of data is it, who stated the data, where and when it was stated, what else was going on in the world when this data was stated, and so forth.[5]

- Contextualization is vital in modifying senseless data into real information - data that can be utilized as useful information that allows intelligent decision-making.
- Contextualization is the system that considers the on-going situation and conditions together with general world knowledge, which also often impacts the decision for or against a certain alternative. Doing the right thing-even if circumstances change. In the end, our overworked minds want spoon-fed insights. We want key-points/catch-words and tightly packaged summaries of relevant, interesting information that will arm us with knowledge and improve our intelligence.[3]
- Context of content: source, media-type, sentiment, demographics, date/time along with the learners' need.

ii. **Collaborative:** It makes predictions conditioned with the tastes and preferences of the learner and allows you to make quality content. Following is the essence of collaborative filtering: 2 users who have liked the same content before will like the one in the future.

B. Natural Language Processing (NLP)

Data is being produced as we speak, as we tweet, as we send messages on whats-app and in various other actions. Most of this data comes in the textual form, which is extremely unstructured. Despite having high dimension data, the information provided in it is not directly available unless it is processed manually (read and understood) or analyzed by an automated system. To generate significant and actionable insights from text data, we can use NLP mechanisms.

NLP is a branch of data science that comprises of systematic processes for understanding, analyzing, and deriving information from the text data efficiently and smartly. By applying NLP and its components, one can organize the vast chunks of text data, perform numerous automated tasks and solve a wide variety of challenges such as - automatic summarization, machine translation, named entity recognition, relationship extraction, speech recognition, topic segmentation, topic labeling, sentiment analysis, and auto-tagging, etc.

The basic high-level workflow of NLP:

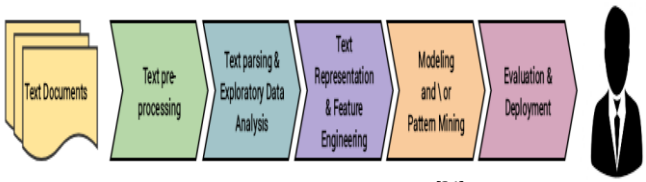


Figure 3: NLP Workflow [24]

a) CORPUS

- The first thing necessary for natural language processing (NLP) tasks is corpus. [10]
- In NLP and linguistics, **corpus** refers to a set of texts or a collection of text documents. It can be thought of as just a bunch of text files in a directory, often alongside many other directories of text files. A corpus is a large and structured set of machine-readable texts that have been produced in a natural communicative setting.
- There's a need to train the model or algorithm with an abundance of data, in the domain of NLP. For this purpose, researchers have assembled many text corpora. Typically, each text corpus is a collection of text sources. There are tons of such corpora for a range of NLP tasks.[11]

A corpus can be assembled from a variety of sources and genres. They can be derived in different ways like:

1. **Web text**: The good thing is that the internet is filled with ext, and in many cases, this text is collected and well organized, even if it requires some finessing into a more usable, precisely-defined format.[10]

2. **Custom Corpus**: We can create a corpus using the NLTK library. NLTK already defines a list of data paths or directories in **nltk.data.path**.[9] Our custom corpora must be present within any of these given paths so it can be found by NLTK.

b) Text Preprocessing

Of all the existing data, textual data is the most unstructured form. Different kind of noise is present in the data and it is not easily analyzable without any pre-processing. The overall process of standardization of text and cleaning, making it ready and noise-free for analysis is called text preprocessing.

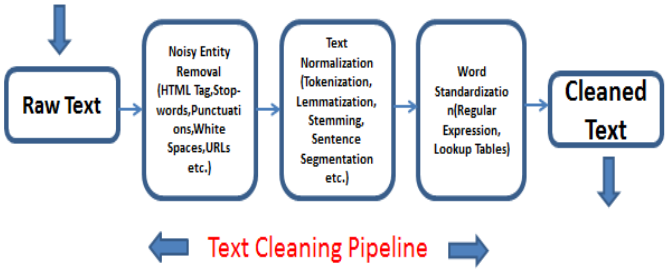


Figure 4: Text Cleaning Pipeline [12]

- **Tokenization:**

Tokenization in NLP is the method by which a huge quantity of text is divided into smaller parts called tokens, and simultaneously throwing away certain characters, such as punctuation. [12]

Given text: Words are out-streaming like limitless rain into a paper cup, they crawl while they pass, they slip away across the universe.

Tokenized form:

Figure 5: Example of tokenization [13]

- **Noise Removal:**

Any piece of content which is irrelevant to the end-output and the context of the data can be determined as the noise.
For example - URLs or links, language stop-words (Most common words of a language - the, is, of, am, in, etc), social media entities (hash-tags, mentions), industry-specific words and punctuations. This step deals with the removal of all types of noisy units present in the text. Also, we will remove punctuation, numbers, and unnecessary white spaces. We use the **nltk** library to discard stop words and libraries to remove punctuation, numbers, and white spaces.

- **Text Normalization:**

Another type of noise is the repetitions by a single word multiple times. For example, run, running, runs, etc are different variations of term run. Normalization will helps reduce such words to a single word. Thus, helps in reducing the dimension. [11] There are mainly two normalization techniques: 1. Stemming
2. Lemmatization

- **Standardization:**

Text data often contains words or phrases which are not present in any standard lexical dictionaries.[13] These pieces are not recognized by search engines and models. E.g - hashtags with attached words, acronyms, and colloquial slangs. Through manually prepared data dictionaries and regular expressions this kind of noise can be fixed;

c) Text to Features (Feature Engineering on text data)

To analyze a preprocessed data, it needs to be converted into features. Depending upon the usage, text features can be constructed using different techniques: Syntactical Parsing, Entities / N-grams / word-based features, Statistical features, and **word embeddings**.
NLP is going to be used by the following block: Text Summarization

C. Text Classification

Text classification can be done in 2 different ways: manual and automatic classification. [16]

Earlier, a human annotator translates the content of the text and categorizes it accordingly. This approach mostly offers quality results but it's time-consuming and costly. The latter applies natural language processing, machine learning, and other methods to automatically classify text in a more cost-effective and faster way.

Text classification with ML makes classifications based on `past observations. By making use of pre-labeled data as training data, a machine learning algorithm can learn the various links between parts of the text and that a specific output (i.e. tags) is expected for a particular input (i.e. text).

The initial stage towards training a classifier with machine learning algorithms is feature extraction. It's an approach used to transform every single text into a vector form (numerical representation). At that point, the machine learning algorithm is fed with training data that comprises of labels/tags (e.g. *sports, education, politics*) and pairs of feature sets (vectors for each text example) to generate a classification model.

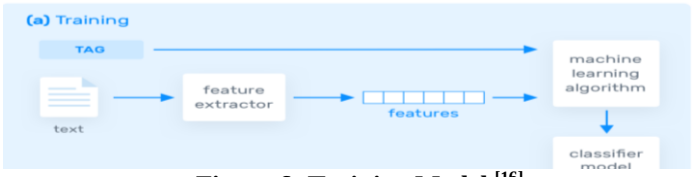


Figure 6: Training Model [16]

Once it's trained with sufficient training examples, the machine learning model can start making accurate predictions. A similar feature extractor is utilized to transform unknown texts to feature sets that could be given to the classifier to obtain predictions on tags (example: sports, research, and politics):

Figure 7: Prediction Model [16]

Text Classification Algorithms:
Some of the best-known machine learning algorithms for creating text classification models involves the support vector machines, deep learning, and naive Bayes' family of algorithms.[16]

a) Naïve Bayes: Gives good outcomes when computational resources are scarce and data available is not much (~ a couple of thousand tagged samples).
b) SVM: It is just one out of many algorithms we can choose from when doing text classification. Like naive Bayes, SVM doesn't need more training data to start giving précised

results. Even though it requires more computational resources than Naive Bayes, SVM can obtain more accurate results.

c) Clustering: Text clustering is grouping a set of unlabeled texts in a manner that texts in a similar group (known as *a cluster*) are a lot more alike to each other compared to those in different clusters. Clustering is an unsupervised learning method which means that it has no labeled data that tags the observations with prior identifiers. Thus the absence of they-label (category information), makes it an unsupervised learning technique.

d) Deep Learning: It is a collection of techniques and algorithms inspired by how the human brain works. Text classification has benefited from the re-emergence of deep learning techniques due to their ability to reach high accuracy with less need for engineered features. Text classification uses two important deep learning approaches:

1. Recurrent Neural Networks (RNN) and
2. Convolutional Neural Networks (CNN)

Deep learning algorithms need a way more training data than traditional machine learning algorithms, which means. it should have at least billions of tagged samples. In contrast, traditional machine learning algorithms like NB and SVM reach a specific limit where adding extra training data doesn't enhance their accuracy. Instead, the more data you feed to the deep learning classifiers they perform better and better.

III. PROPOSED WORK

Data set:

Scraped Long Text documents from various websites for training. Each document should consist of:

1. id: a unique identifier for the article
2. title: the title/heading of the article
3. content: the textual content of the article (may occasionally contain HTML & JavaScript content)
4. source: the name of the article source (e.g. Reuters)
5. published: the publication date of the article
6. media-type: either "Articles" or "Blog "or "Book"

I am using the Kaggle-dataset of medium articles [30]:
1 file – 6 Columns

Detail Compact Column 6 of 6 columns

author	claps	reading_ti...	link	title	text
Justin Lee	8.3K	11	https://medium.com/swlh/chatbots-were-the-next-big-thing-what-happened-5fc49dd6fa61?source=---------...	Chatbots were the next big thing: what happened? – The Startup – Medium	Oh, how the headlines blared: Chatbots were The Next Big Thing. Our hopes were sky high. Bright-eyed...
Conor Dewey	1.4K	7	https://towardsdatascience.com/python-for-data-science-8-concepts-you-may-have-forgotten-i-did-82596...	Python for Data Science: 8 Concepts You May Have Forgotten	If you've ever found yourself looking up the same question, concept, or syntax over and over again w...
William Koehrsen	2.8K	11	https://towardsdatascience.com/automated-feature-engineering-in-python-99baf11cc219?so	Automated Feature Engineering in Python – Towards Data Science	Machine learning is increasingly moving from hand-designed models to automatically

Figure 8: Description of Dataset

Architecture:

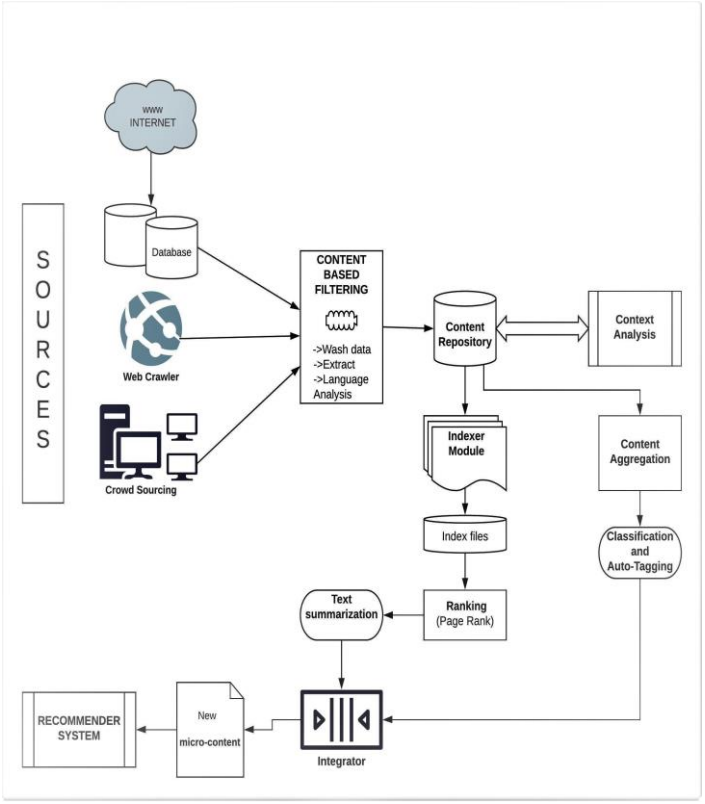


Figure 9: System Architecture

This proposed model will have some major modules, which are:

A. Sources

Web Crawling, Web Scrapping

It downloads the content from all over the Internet and indexes it**.** A web crawler bot is like someone who goes through all the books in an unorganized library and gathers a card catalog such that anyone who visits the library can easily and quickly get the information they need. [26] To help, sort and categorize the library's books by topic, the organizer will read the title, summary, and some of the inside text of every book to find out what it's about.

B. AI-based filtering

Not all the sourced content is relevant, so we set up filters.

Millions of readers –> like, share, post. So, they generate billions of data points. So by analyzing the experiences/interests of the learners and by using this data to find patterns of what resonates across audiences – we give it to the AI engine – AI takes all the data and turns it into actionable information, telling you exactly what courses or contents the learners need to take. AI-based filtering will analyze the content of documents, articles, books, pdf, etc as well as considers the context of the content and then will store it to the content-repository.

We will curate the content by applying AI-based content filtering and Context-based filtering algorithms

C. Content-Repository:

All the filtered content will be stored in this content repository.

Text data from the internet comes in tonnes and various forms: CSV files or unstructured data, thus managing, store, analyze and make this huge amount of digital information available to learners is a challenging task. Digital information is highly dynamic (the original contents and associated metadata keeps getting changed based on the needs) and bulky (the size of the content varies, ranging from a few kb's to Tb's and maybe more!). [14] To efficiently handle these requirements big-data solution like Hadoop is used.

D. Indexing:

The content in the repository needs to be indexed and ranked.

Indexing - collects, parses, and stores the content to facilitate accurate and fast **information retrieval**. The goal of storing an index is to optimize performance and speed in finding relevant content for a search request. Without an index, it would **scan** each document in the **corpora**, which would need significant computing power and time. For instance, while an index of 10,000 documents can be queried within milli-seconds, a sequential scan of each word in 10,000 large documents could go on for hours.

When you have a huge amount of content you need a way to shortcut to that content. We can't just have one big database containing all the pages, which they organize through, whenever a query is fired. It would be way too slow. Rather, they create an index which essentially shortcuts this process. Thus, we will use technology such as Hadoop to manage and query vast amounts of data very fast. Searching the index is far quicker than searching the entire database each time.

Common words such as 'and', 'the', 'if' are not stored. These are known as stop words. They don't generally add to the search engine's interpretation of the content (although there are exceptions: "To be or not to be" is made up of stop words) so they are removed to save space. It might be a very small amount of space per page, but when dealing with billions of pages it becomes an important consideration. This kind of thinking is worth bearing in mind when trying to understand Google and the decisions it makes. A small per page change can be very different at scale.

E. *Ranking:*

Ranks the pages based on what it thinks the learner wants.

Common words such as 'and', 'the', 'if' are not stored. These are known as stop words. They don't usually include the search engine's interpretation of the content (despite there are exceptions: "To be or not to be" consisting of stop words) so they are eliminated to maintain space. It might be a really small amount of space of every single page, but when dealing with millions of pages it becomes an important factor. This kind of thinking is worth bearing in mind when trying to understand Google and the decisions it makes. A small per page change can be very different at scale.

Page Rank Algorithm:

PageRank (PR) is an algorithm used by Google Search to rank webpages in its search engine outcomes. It is an approach to determining the importance of web pages. The significance of a Website page is an inherently subjective thing, which relies on the readers' knowledge, attitudes, and interests. But there is still more that can be said objectively regarding the relative importance of web pages.[27] PageRank, rates Web pages mechanically and objectively, efficiently measuring the human attention and interest devoted to them.

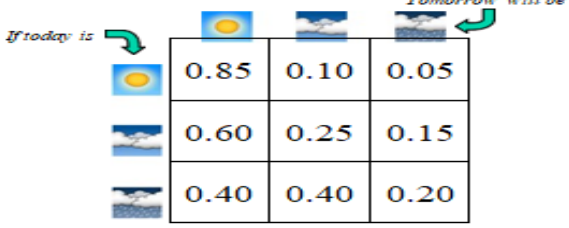
The PageRank algorithm provides every page a rating of its importance, which is a determined measure by which a page becomes noted if important pages link to it[27]. This definition is recursive because the importance of a page refers back to the different pages that link to it.

PageRank can be thought of as a random surfer on the Internet, following links from page to another. The page rank of any page is approximately the possibility that the random surfer will come on a particular page.

Whereas more links visit the important pages, the surfer is more likely to end up there. The behavior of the random surfer is an example of a Markov process, which relies only on the current state of a system and not on its history.

Markov Model: Google's random surfer is an example of a Markov process, in which a system moves from state to state, based on probability information that shows the likelihood of moving from each state to every other possible state [27].

possible state.



STEPS OF PAGE RANK ALGORITHM[25]:

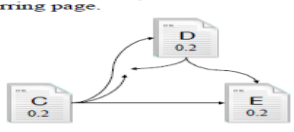
1. Start with a set of pages.

2. Crawl the web to determine the link structure.

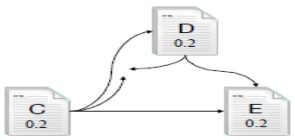
3. Assign each page an initial rank of 1/ N.

4. Successively update the rank of each page by adding up the weight of every page that links to it divided by the number of links emanating from the referring page.
   - In the current example, page E has two incoming links, one from page C and one from page D.
   - Page C contributes 1/3 of its current page rank to page E because E is one of three links from page C. Similarly, page D offers 1/2 of its rank to E.
   - The new page rank for E is

$$PR(E) = \frac{PR(C)}{3} + \frac{PR(D)}{2} = \frac{0.2}{3} + \frac{0.2}{2} = 0.17$$

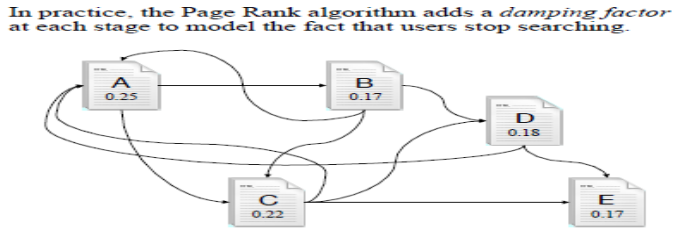


5. If a page (such as E in the current example) has no outward links, redistribute its rank equally among the other pages in the graph.
   - In this graph, 1/4 of E's page rank is distributed to pages A, B, C, and D.
   - The idea behind this model is that users will keep searching if they reach a dead end.



6. Apply this redistribution on every page of your graph.

7. Repeat this process until the page ranks stabilize



F. Text-Summarization

Microlearning can take a variety of forms like Texts, games, podcasts, videos, PowerPoint's. Research indicates that short microlearning contents are the most engaging and effective medium. Here we will perform microlearning that will focus on generating micro-text using Text summarization Algorithm, which will break the content down into easily digestible, short components (typically 60-80 words in length) that address a single learning objective.

- Automatic Text Summarization is amongst the most interesting and challenging problems in the area of NLP.
- It is a technique of generating/creating a short, accurate, concise, meaningful, and fluent summary of a longer text document from several different text resources like articles, books, blog posts, news, research papers, tweets, and emails.
- Automatic text summarization techniques are highly needed to tackle the ever-growing amount of text data available on the Internet both-- to consume relevant information faster and better help discover relevant information.
- There are many reasons and uses for a summary of a larger document:

- Headlines (from around the world)
- Outlines (notes for the student)
- Employees (on-time learning)
- Minutes (of a meeting), and many more.

Automatic text summarization methods are greatly needed to address the ever-growing amount of text data available .i.e. to say the most important things in the shortest amount of time.

- **Abstraction-based Summarization:**

- When a human is given a text corpus to summarize, they might rephrase the key points in their own words. This is known as abstractive summarization and it needs high-level human skills like the ability to combine multiple perspectives into coherent natural language.
- These methods use advanced NLP techniques to generate an entirely new summary. Some parts of this summary may not even appear in the original text.
- **Algorithm**: **Encoder-decoder Model with an attention mechanism.**

Step 1 - Tokenization:

- Tokenization in NLP is the process by which big quantity of text is divided into smaller parts called tokens.
- Machine learning models need numeric data to be trained and make a prediction.
- Word tokenization becomes an important part of the text (string) to numeric data conversion i.e Glove

```
from collections import Counter
def get_vocab(combinedText):
    #  to get vocab and count in another way,
    words = combinedText.split()
    vocab = [word for word, word_count in Counter(words).most_common()]
    return vocab,words

vocab,words = get_vocab(heads[i]+descs[i])
```

```
print (vocab[:50])
print ('...', len(vocab))
```

```
['the', 'to', 'a', 'and', 'of', 'that', 'is', 'with', 'are', 'in', 'The', 'bot', 'we', 'or', 'were', 'for', 'And', 'an', 'at', 'can',
'but', 'bots', 'was', 'they', 'on', 'than', 'more', 'as', 'how', 'it', 'chatbots', 'be', 'just', 'you', 'it's', 'from', 'will', 'what',
'not', 'even', 'apps', 'user', 'by', 'we're', 'information', 'only', 'would', 'take', 'up', 'hype']
... 1035
```

Step 2 - Word Embeddings:

- A word embedding is a real number, which is a vector representation of a word.
- Word embeddings are a type of word representation that can be words with the same meaning to have the same representation.
- The objective is to capture some kind of relationship in that place, be it morphology, meaning, context, or some other sort of association.

Step 3 - GloVe: Global Vectors

The GloVe is a word vector method, in which word vectors place words to a fine vector space, where the same words cluster together and different words repel.

```
! pip install -q tqdm flair
import numpy as np
from tqdm import tqdm
glove_name= ! wget "http://nlp.stanford.edu/data/glove.6B.zip" -O glove.6B.zip && unzip glove.6B.zip

word2vec = {}
with open('glove.6B.300d.txt','r', encoding='utf-8',errors='ignore') as f:
  for line in tqdm(f, total =400000):
    #glove_n_symbols = line.decode().split(b":")[1]
    glove_n_symbols = line.split()
    word = glove_n_symbols[0]
    #coefs = np.fromstring(glove_n_symbols, "f", sep=" ")
    #coefs = np.asarray(glove_n_symbols[1:], dtype='float32')
    coefs = np.asarray([float(val) for val in glove_n_symbols[1:]])
    word2vec[word] = coefs

print('Found %s word vectors.' % len(word2vec))
#http://nlp.stanford.edu/data/glove.42B.300d.zip
```

The corpus will generate a bigger vector which is represented graphically:

```
100%|          | 400000/400000 [01:03<00:00, 6276.84it/s]

Found 400000 word vectors.
```

Vectors:

```
print(coefs)

[ 0.429191   -0.296897    0.15011    0.245201  -0.00352027 -0.0576971
  0.1409     -0.222294    0.221153   0.767218  -0.0772662  -0.0710635
  0.0629486  -0.220179   -0.108197  -0.301419   0.232164    0.168669
 -0.00452476  0.168254   -0.0579106 -0.0362662 -0.273464   -0.162976
  0.239398   -0.0119058   0.044685   0.105252   0.102867   -0.0232984
 -0.0114432  -0.381673    0.06122    0.0170547  0.415463   -0.109101
  0.0959916   0.19149    -0.00752907 -0.194603  -0.0431976   0.259788
  0.00527856 -0.183626    0.225188  -0.0187726 -0.158172   -0.586937
  0.249259   -0.130252   -0.0537497  0.0315535 -0.18562     0.0610198
 -0.0850566  -0.0965162   0.278621  -0.247254  -0.153895    0.0418453
  0.0704212  -0.062286   -0.284913   0.0152124  0.144002    0.335902
 -0.288315   -0.00253548 -0.0876423 -0.0574409  0.00670068 -0.0753335
 -0.0677815  -0.056624    0.19296    0.0250159 -0.39188    -0.159278
  0.26123     0.10221     0.0877169  0.0433055 -0.179803   -0.189744
  0.0510538  -0.0164141  -0.00714073 -0.327697  -0.207509   -0.0213479
  0.116692   -0.0675631   0.268143   0.0961855  0.0516012  -0.0365261
  0.317162   -0.158929   -0.055459   0.287867  -0.140655   -0.22574
 -0.0546181   0.212033   -0.0359359 -0.0979935 -0.0192465  -0.186423
  0.298623   -0.133734   -0.114258   0.303311   0.142693    0.0511059
  0.111157   -0.106419    0.246942  -0.0651711  0.137669    0.227577
```

Step 4 – Apply Algorithm: Encoder-Decoder Model with Attention Mechanism

Let's consider a simple example to understand how Attention Mechanism works:
Source sequence: "Which sport do you like the most?
Target sequence: "I love cricket"

The first word 'I' in the target sequence is connected to the fourth word 'you' in the source sequence, right? Similarly, the second-word 'love' in the target sequence is associated with the fifth word 'like' in the source sequence. So, rather than searching for the words in the source sequence, we can improve the value of specific parts of the source sequence that result in the target sequence. This is the primary idea behind the attention mechanism. [28]

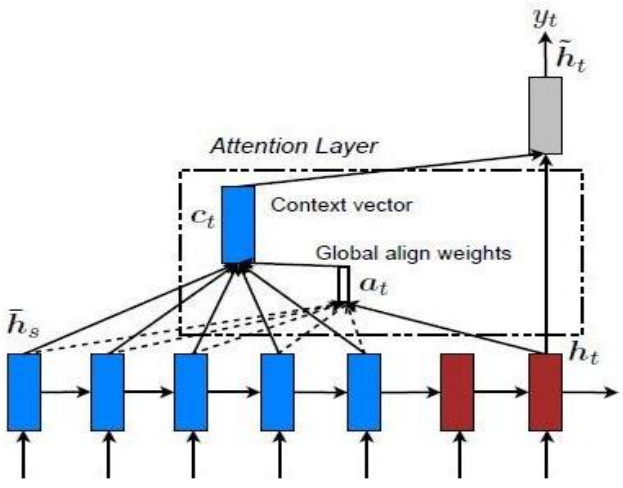


Figure 10: Working of Attention Model [19]

How attention mechanism work:

- The encoder outputs the hidden state (h_j) for each time step j in the source sequence.
- Likewise, the decoder outputs the hidden state (s_i) for each time step I in the target sequence.
- We compute a score called an alignment score (e_{ij}) based on which the source word is aligned with the target word using a score function. The alignment score is computed from the target hidden state s_i and the source is hidden state h_j using the score function. This is determined by $\mathbf{e_{ij}}$**= score (**$\mathbf{s_i}$**,** $\mathbf{h_j}$**)**
- Where $\mathbf{e_{ij}}$ denotes the alignment score for the target timestep I and source time step **j**.
- We normalize the alignment scores using the softmax function to fetch the attention weights ($\mathbf{a_{ij}}$):

$$a_{ij} = e^{eij} / \sum_{k=1}^{Tx}$$

- We calculate the linear sum of products of the hidden states of the encoder $\mathbf{h_j}$ and the attention weights $\mathbf{a_{ij}}$ to generate the attended context vector ($\mathbf{C_i}$):

$$C_i = \sum_{j=1}^{Tx} a_{ij}$$

- The target is a hidden state and attended context vector of the decoder at timestep **i** are joined to create an attended hidden vector $\mathbf{S_i}$.

$$S_i = \text{concatenate}([s_i; C_i])$$

- S_i is then fed into the dense layer to produce y_i

$$y_i = \text{dense}(S_i)$$

The output generated is a summary that we call a micro-content.

***G. Content Aggregation*:**
Text classification and auto-tagging - The content in the content repository needs to be curated and aggregated using Text clustering [17]

Content aggregation is simply curating. In other words, it is the collection of relevant data covering the same topic, and displaying it all in one place for easy access and reference by users. [15]

The content in the content repository needs to be curated and aggregated using Deep Learning Algorithms. This is similar to a single large search engine where only essential information gets collected while all irrelevant material or of low-quality is weeded out.
Content comes in different formats so we need to aggregate.
We can do this by following two steps:
1. Extracting Tags
2. Categorize/Classify text using Text Classification

Text classification is the method of assigning categories or tags to text according to its content. It's one of the crucial tasks in NLP with wide applications like topic labeling, sentiment analysis, intent detection, and spam detection.

Unstructured data in textual form is everywhere: blogs, emails, articles, social media, web pages, survey responses, support tickets, and more. Text can be a highly rich source of information; however, extracting insights from it can be

time-consuming and hard because of its unstructured behavior. Corporate sectors are moving to text classification for structuring text in a cost-efficient and fast way to improve decision-making and automate processes.

Text classification (commonly known as *text tagging or text categorization*) is the role of assigning a set of predetermined categories to free-text. Text classifiers can be utilized to categorize, organize, and structure, almost anything. For example, favorite online content curated into bookmarked folders, brand mentions can be curated by sentiment, new articles can be curated by topics, chat conversations can be organized by language, and support tickets can be organized by urgency, and so on.

Algorithm:

Today there are vast heaps of data and unfortunately, most of it is unstructured. There is an abundance of data in the form of free-flow text residing in our data repositories. While there are many analytical techniques in place that help process and analyze structured (i.e. numeric) data, fewer techniques exist that are targeted towards analyzing natural language data.

The algorithm first performs a series of transformations on the free flow text data (elaborated in subsequent sections) and then performs a k-means clustering on the vectorized form of the transformed data. Subsequently, the algorithm creates cluster-wise tags, also known as cluster-centers that are representative of the data contained in these clusters.
The solution boasts of end-to-end automation and is generic enough to operate on any dataset.
The text clustering algorithm works in five stages enumerated below: -

1) Transformations on a raw stream of free flow text:
2) Creation of Term Document Matrix
3) TF-IDF (Term Frequency – Inverse Document Frequency) Normalization
4) K-Means Clustering using Euclidean Distances
5) Auto-Tagging based on Cluster Centers

Example:

Take a look at the following text: ***"The user interface is easy to use and quite straightforward."***

A classifier takes text as an input, analyzes its content, and then automatically assigns relevant tags, like *UI* and *Easy to Use* that represent this text:

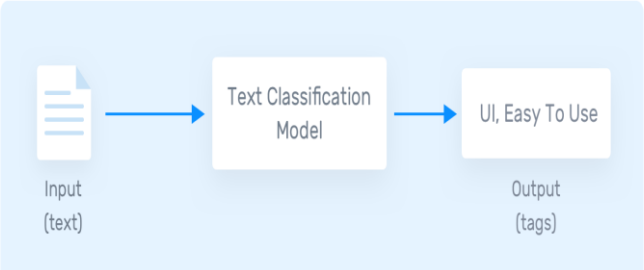


Figure 11: Auto Tagging Model [16]

H. Integrator: It will have:

1. Original content with tag –> auto-tagging –take the article →tag and classify it.
2. Micro-content –> summarized text.

Depending on the tag you're requesting integrator will pull that tag and its corresponding summary and generate a New content -> summarized article + tag which is then given to the recommender system.

IV. RESULT AND ANALYSIS

Because of the limited resources and data, this is the only thing I could implement. A laptop is not sufficient to build the model I'm trying to build. The model is mathematically complicated and it takes a lot of data to train the model and then test it. As the size of the content grows, it becomes challenging to train the model to summarize it in limited words. For that, you need a large amount of data to train the model and test the model. Also, with a lot of data, you need a lot of processing power to process that data.

Following are the challenges which I came across:

A. CHALLENGE 1: Needs more data

Why NLP models take/need a lot of data to train the model?

"One of the biggest challenges in natural language processing is the shortage of training data. Because NLP is a diversified field with many distinct tasks, most task-specific datasets contain only a few thousand or a few hundred thousand human-labeled training examples." – Google AI [23]

A lot of the problems around locking in a target lot of data points originate from the purpose of the training process. The objective of the training is to make a model understand the relationships and patterns behind the data, instead of the data itself. When collecting data, you should be certain that you have sufficient to give your algorithm an accurate picture of the complex network of meaning that exists between and behind your data points. This might seem like a straightforward practice on the surface. However, the diverse goals of projects can result in a vast range of training data types. Thus, every project has a unique combination of factors that make it highly challenging to work out your data needs ahead of time. These may include some or all of the following [23]:

- **The complexity of the model: The complexity of the model:** Each parameter that the model has to consider for the purpose to perform its task increases the volume of data that it will require for training. , a model which identifies the making of a particular car has a small, set number of parameters relating mostly to the shape of the vehicle. A model that has to determine how much that car costs have a much greater picture needs to understand, including not only the make and condition of the car but also economic and social factors. Thanks

to this greater degree of complexity, the second model will need significantly more data than the first.

- **Training method:** As models are forced to understand a vast number of interlinking parameters, the resulting complexity forces a change in the way that they are trained. Traditional machine learning algorithms use structured learning, which is to say that they quickly reach a point where additional data has very little ROI. Whereas, deep learning models find out their parameters and learn how to improve without structure. This indicates that they not only require significantly more data whereas also have a much longer learning curve where further data has a positive impact. As an outcome, the training process you use will cause significant variation in the amount of training data that is useful to your model.

- **Labeling needs:** Depending on the task you're doing; data points can be labeled in different ways. This can cause a significant difference in the number of labels your data produces, in addition to the effort it takes to create those labels. For instance, if you have 1000 input sentences of data for sentiment analysis, you may only need to label it as negative or positive and as a result, it produces one label per sentence. However, if those same 1000 sentences are annotated for feature extraction, there's a need to label 5 words per sentence. Despite having the same raw input data, one task yields five times more labels than the other. The way you train your data can, thus, affect the amount you need for your project, as well as the price of procuring it.

- **Tolerance for errors:** The intended role of the model within your business also affects data quantity. A 20% error rate is acceptable for a model that predicts the weather, but not for one that detects patients at risk of heart attacks. Improvement of edge cases is what will reduce this risk. If your algorithm is highly risk-averse or integral to the success of your business, the amount of data you need will increase to reflect the need for accurate performance.

- **Diversity of input:** In this complex world, the model is fed with a variety of inputs. For example, a chatbot should be able to understand a variety of languages, written in a range of informal, formal, and even grammatically incorrect styles. In cases where your model's input won't be highly controlled, more data will be necessary to help your model function in that unpredictable environment.

From this, it's clear to see that the amount of data you need is decided by your project's unique needs and goals.

B. CHALLENGE 2: Compute Intensive Hardware

Why do we need more hardware for deep learning?

The smaller your system more is the more time you will require to make a trained model that performs well enough. Deep learning is an **algorithm** - a software construct. [29] We describe an artificial neural network in our preferred programming language which would then be converted into a set of commands that run on the computer.

If you would have to figure out which components of the neural network do you think would require intense hardware resource, what would be your answer?

A few candidates from the top of my mind are [29]:

- Training the deep learning model
- Storing the trained deep learning model
- Deployment of the model
- Preprocessing input data

Amongst them, training the deep learning model is the most intensive task.

Expected Results:

Curator: Curator filters, tags, and classifies the text/article.
Encoder–Decode-Attentional model: Summarizes the given text into micro-form.

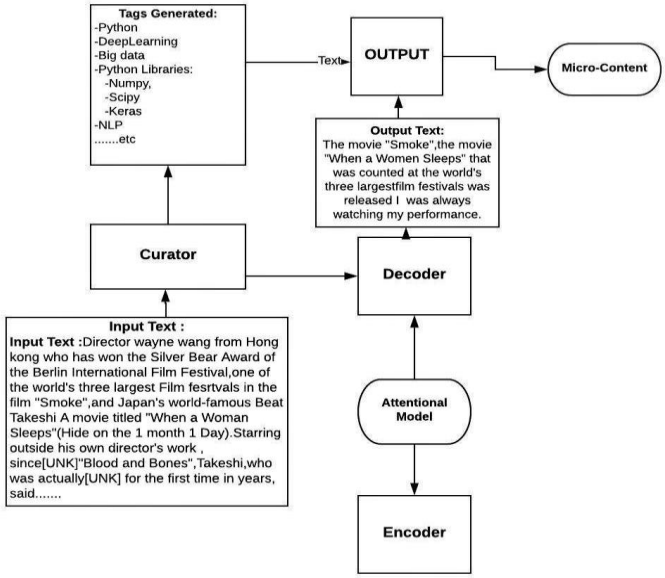


Figure 12: Results

Conclusions and future scope

The main focus of this paper is to build a platform that provides content management, knowledge management, and learning systems all in one. Through the principles of micro-learning using AI-based Learning Object curation, the system is focusing on skill-based learning. The learner is suggested with content specifically targeted to the specific domain based on their needs, different levels of topic knowledge, experience, and interest in microform. It appeals to learners to get the micro-content as it consumes less time and is available to them exactly at the time of the learning need (just-in-time).

This research focused on generating the micro-content of text but the future aspects of research will focus on generating and curating micro-videos.

REFERENCES

[1] KAKLIJ, V.A., SHAH, M.K.V., AND MANDAWKAR, M.U., 2019. MICROLEARNING BASED CONTENT-CURATION USING ARTIFICIAL INTELLIGENCE FOR LEARNING EXPERIENCE PLATFORM: A SURVEY. *IJRAR-International Journal of Research and Analytical Reviews (IJRAR)*, *6*(4), pp.580-584.

[2] Bersin, J., 2018. A new paradigm for corporate training: learning in the flow of work. Insights on Corporate Talent, Learning, and HR Technology.

[3] https://www.wired.com/insights/2013/04/with-big-data-context-is-a-big-issue/

[4] https://www.nuance.com/content/dam/nuance/en_us/collateral/mobile/automotive/white-paper/wp-personalization-en-us.pdf

[5] https://www.smarthint.co/en/ai-product-recommendation-engine/

[6] https://www.craft.ai/blog/building-a-news-recommendation-application-based-on-a-user-preferences-and-context-with-craft-ai

[7] https://www.geeksforgeeks.org/nlp-custom-corpus/

[8] https://www.aclweb.org/anthology/E06-1030.pdf

[9] https://devopedia.org/text-corpus-for-nlp

[10] https://www.kdnuggets.com/2017/11/building-wikipedia-text-corpus-nlp.html

[11] https://medium.com/@arunm8489/getting-started-with-natural-language-processing-6e593e349675

[12] https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1

[13] https://www.analyticsvidhya.com/blog/2017/01/ultimate-guide-to-understand-implement-natural-language-processing-codes-in-python/

[14] http://www.devx.com/enterprise/big-data-technologies-content repository.html

[15] https://curator.io/blog/ultimate-guide-to-content-aggregators

[16] https://monkeylearn.com/text-classification/

[17] https://medium.com/analytics-vidhya/k-meanception-how-to-automatically-tag-news-articles-using-clustering-algorithms-7357f18031e2

[18] https://www.kdnuggets.com/2017/06/text-clustering-unstructured-data.html

[19] https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/

[20] https://arxiv.org/abs/1602.06023

[21] https://www.youtube.com/watch?v=ogrJaOIuBx4

[22] https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/

[23] https://lionbridge.ai/articles/how-much-ai-training-data-do-you-need/

[24] https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72#:~:text=Standard%20NLP%20Workflow&text=We%20usually%20start%20with%20a,using%20relevant%20feature%20engineering%20techniques

[25] Rai, P., and Lal, A., 2016. Google pagerank algorithm: Markov chain model and hidden Markov model. *International Journal of Computer Applications*, *138*(9), pp.9-13.

[26] https://www.cloudflare.com/learning/bots/what-is-a-web-crawler/

[27] https://cs.stanford.edu/people/eroberts/courses/cs106b/handouts/53-MoreAlgorithms.pdf

[28] https://towardsdatascience.com/text-summarization-from-scratch-using-encoder-decoder-network-with-attention-in-keras-5fa80d12710e

[29] https://www.analyticsvidhya.com/blog/2017/05/gpus-necessary-for-deep-learning/

[30] https://www.kaggle.com/hsankesara/medium-articles