

Repurpost

In association with,



REPURPOST

WEB APPLICATION FOR AN AUTOMATIC TAGS GENERATION SYSTEM

PROJECT REFLECTION PAPER

Key Stakeholders

Name	Title	Project Role
Alexandra Cowen	Chief Executive Officer (CEO)	Sponsor
David F.	Chief Technology Officer (CTO)	Co-Sponsor
Tamara Fromosa	Chief Experience Officer (CXO)	Technical Advisor
Yashwanth Balan Arumugam	Intern	Project Lead and Coordinator
Pragya Avinash Mishra	Intern	Research and Technical Support
Atharva Shantanu Kulkarni	Intern	Research and Technical Support
Shruti Sham Kotwal	Intern	Architecture Design & Development Support
Saju Chacko Rajan	Intern	Research, Quality and Development Support

Table of Contents

S. No.	Title	Page No.
1.	Executive Summary	4
2.	Introduction	5
3.	Technical platforms, Tools, and Dataset	6
4.	Implementation and High Level Architecture and Design	8
5.	AI Models Implementation Process	14
6.	Establish Codebase Repository	17
7.	Web API and Application Development	18
8.	Discussion	21
9.	References	28

1. Executive Summary:

Repurpost has at its core a content repository (DAM) where users can classify their content based on strategy (campaign, buyer persona, product line) or by tags. Repurpost team has identified that the users frequently do not know the right tags and provide incorrect tags. This results in content being mis-classified and or tags not useful for other platform users.

1.1 Objective:

The main objective of the Repurpost project is to perform a proof-of-concept web application that can auto suggest tags and content classification based on the content text that is provided by the users on the Repurpost platform. This will allow the content on the platform to be better classified, allow more search options and help the users in creating focus content based on tag trends.

1.2 Problem Statement and Solution:

The solution to this requirement is provided by integrating a new web API with the existing Repurpost web application. In order to provide this solution, the steps followed were:

1. Identified a similar dataset that can be used to train a machine learning model
2. Trained a Multi-label classification NLP model that receives text input and provides tag suggestions.
3. Created and deployed a web API that runs the trained multi-label classification model.
4. Created a web application to mimic Repurpost platform.
5. Integrate API calls with web application and display tag suggestions

With the provided solution, content creators will get tag suggestions with a **65%** accuracy based on the dataset that was used for training the multi-label classifier. This will help offering better platform experience to the Repurpost users and avoid misclassification of textual content.

This solution implementation is important and proves that a multi-label classification model is a solution to the problem statement. A model that is trained with real Repurpost client text content can provide better accuracy allowing a complete hands-off tagging in the future of the platform.

2. Introduction:

Repurpost has a platform that enables users to produce content there before publishing it to other online and social media platforms. By doing so, a content repository (DAM) is created, which can be used to categorize user content according to strategy (campaign, buyer persona, product line), or by tags. The Repurpost team has discovered that users frequently supply incorrect tags because they are unfamiliar with the proper ones. This leads to incorrectly classified content and/or tags that are useless to other platform users.

Building a web application that can automatically recommend tags and content classification based on the textual material that is submitted by users on the Repurpost platform is the major goal of the Repurpost project. This will make it easier for users to choose the right tags for their content and enable the Repurpost platform to classify and propose content tags for both new and seasoned platform users with greater accuracy. More search possibilities will be available as the platform's material is better categorized, and users will be assisted in producing targeted content based on tag trends.

In addition to using Trello for project management, the project team and the Repurpost sponsor arranged up meetings on Zoom or teams and established Slack channels for ongoing contact.

2.1 Technology Introduction:

Classification issues can be divided into three categories: binary classification, multi-class classification, and multi-label classification. Repurpost's web interface for creating content allows users to assign multiple tags to a piece of content. The architecture and solution call for using a multi-label classification model, in which each tag denotes a different classification issue. However, because the tags are somehow connected, it is possible to suggest multiple tags for the same textual content. The Repurpost team investigated various cloud deployment platforms for web API and web applications, such as vercel.com, render.com, Heroku.com, and Google Cloud, for the suggested solution. Additionally, we experimented with a variety of ML models, including Linear SVC, SGD Classifier, Logistic Regression, BERT Classifier, CNN, and RNN Classifiers.

3. Technical platforms, Tools, and Dataset:

Technical platforms and tools:

Following are the **technical platforms and tools** used for the solution:

- Python – primary programming language.
- Flask – web framework for developing and packaging web applications using python.
- Render.com – cloud platform to deploy flask applications.
- Retool.com – cloud platform for low code we application development.

Communications Tools:

Following are the communications tools used for communicating with the internal and sponsor team,

- Slack – primary communication tool with the sponsor and the technical team.
- Trello – dashboard for maintaining a direct work flow of the progress.
- Google Meet – primary communication tool for meetings and discussion with the internal team members.
- Zoom Meeting – primary communication tool for meetings and discussion with the sponsor and technical team members.

Dataset Summary:

Since real client data from the Repurpost platform cannot be used, a different dataset that is similar to the content data from Repurpost had to be found. The team was able to locate a dataset with multi-label tags, a title and body field that are distinct from one another, and a dataset that will allow the ML model to be trained in close alignment with Repurpost content data.

The identified dataset is Stack Overflow Questions and Tags. 1.25 million features make up the 1.7 GB-sized dataset. Before using NLP algorithms on the dataset, some preprocessing and cleaning must be done will be explained in the implementation section.

4. Project Scope Implementation:

This section serves as a journey map for the project's implementation process, from the creation of the project plan to the deployment of the finished product.

4.1 Project Plan:

A thorough and in-depth project plan is essential for the team to follow in terms of structure and deadlines. This enabled the team to concentrate on the proper work items according to the pre-established plan. The team successfully followed the project plan and was able to deliver each artifact by the deadlines.

4.2 Dataset Research and Identification:

There are numerous datasets that only have one tag or categorization category assigned to a text. These datasets are only appropriate for problems requiring multi-class categorization. The dataset needs information with many tags for the same text content in order to perform a multi-label classification (Deutschman, 2019) approach. The [StackSample: 10% of Stack Overflow Q&A](#) (Stackoverflow, 2019) that satisfies the data requirements was found after some research was done for such a dataset. Although the dataset includes Questions, Tags, and Answers, just the data in the Questions.csv and Tags.csv files must be taken into account and combined in order to meet our requirements.

4.3. High Level Architecture and Design Diagrams:

4.3.1 Architecture Diagram:

The design and architecture for this solution is a common pattern that is followed for web applications and web APIs. The main difference in this architecture is that the web API runs a pretrained multi-label classification model that provides tag suggestions based on user text content in the web interface.

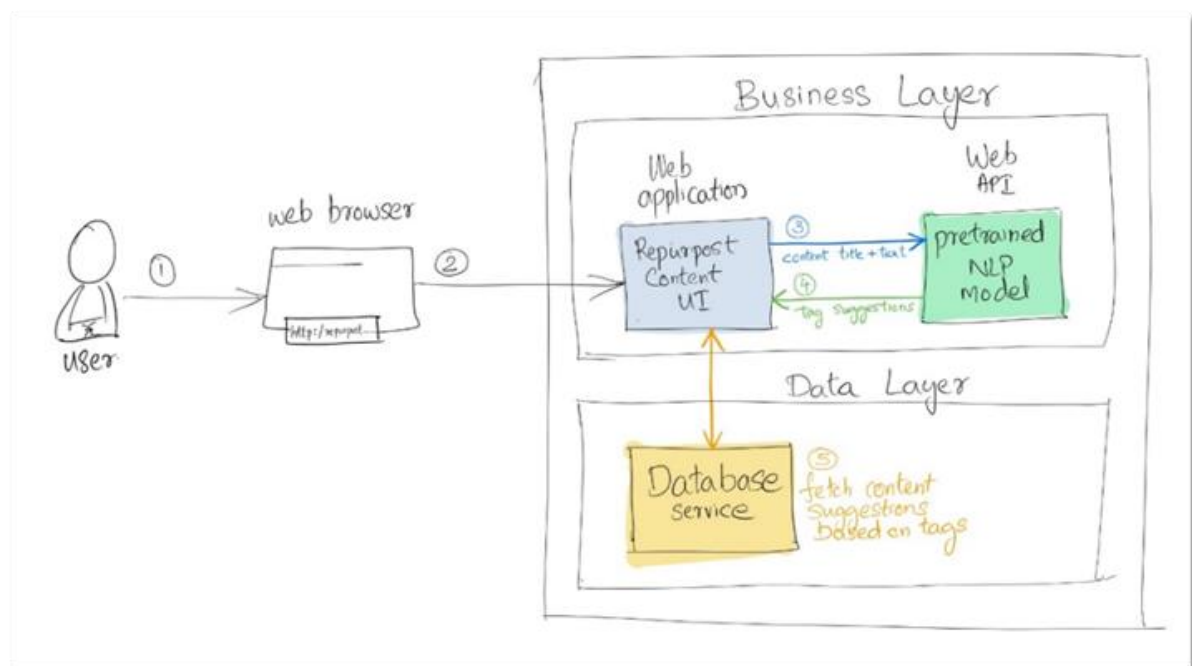


Fig1. High Level Architecture Diagram

4.3.2 User Flow Diagram:

The Repurpost team performed research on the Repurpost platform and identified the current user flow. Based on the current user flow and feedback received from sponsor and professor, the tag suggestions will be displayed on the main page itself. Creating a user flow or user journey allowed the team to create wireframes that was the foundation to developing the web interface for the web API call.

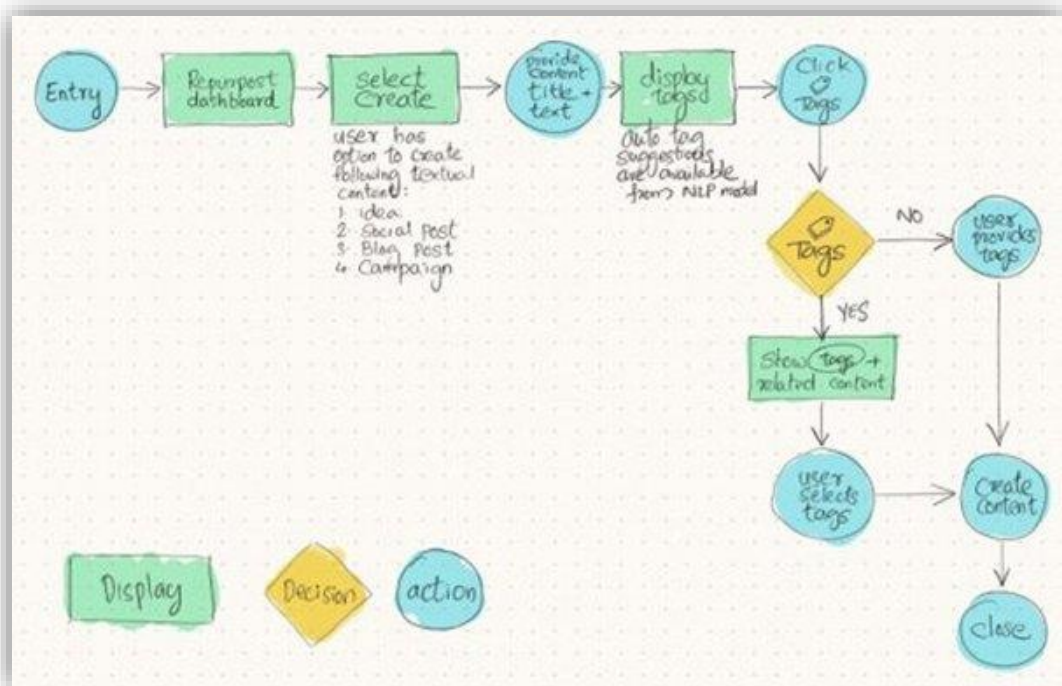


Fig2. User-flow Diagram

4.3.3 User Interface Wireframe Diagram:

Wireframes (Proto.io, 2017) are sketch skeletons of an interface that provides a quick and easy way to visualize a user interface and identify any issues and receive immediate sponsor and stakeholder feedbacks. The feedback that we received from the sponsor and professor was very encouraging and we modified the web interface based on the feedback we received during the initial walkthrough.

Wireframe 1: Content Classification Dashboard

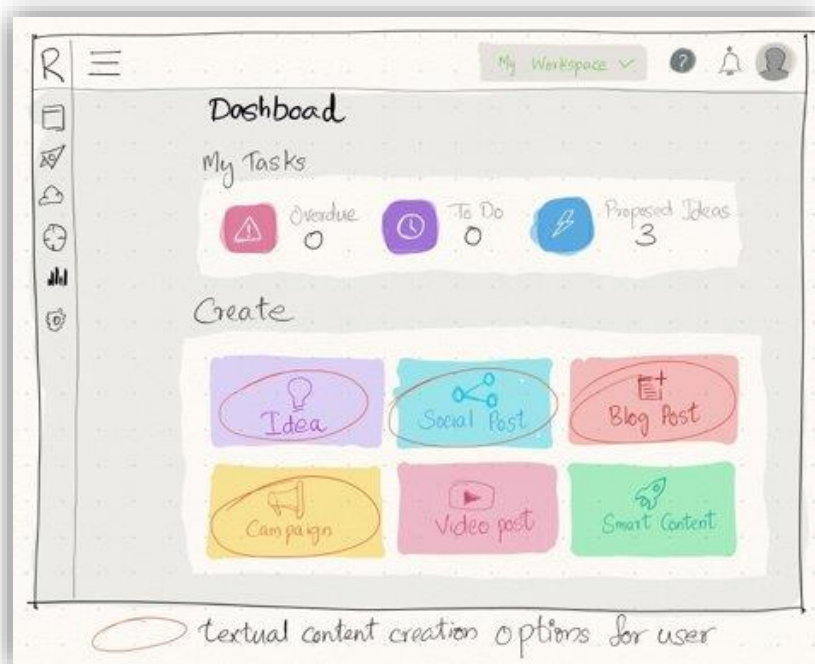


Fig3. Content Classification Dashboard

Wireframe 2: Tags Generation Page

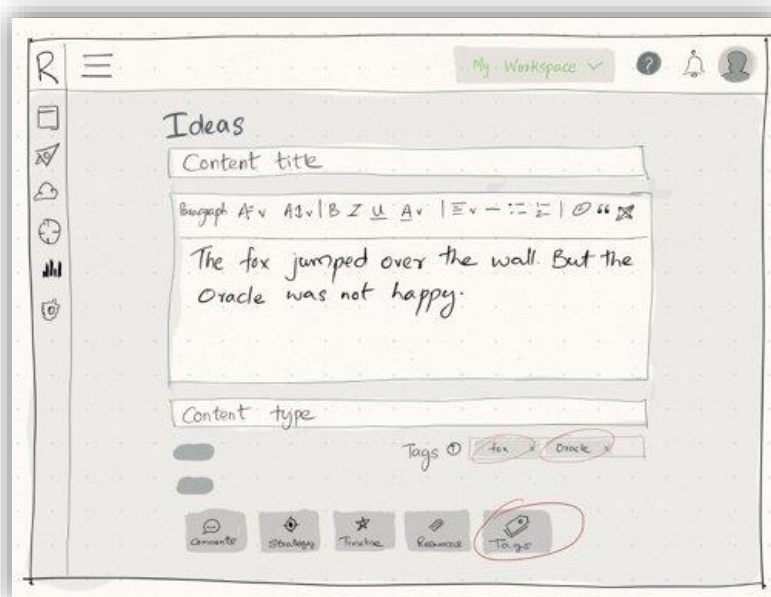


Fig4. Tags Generation UI

4.4 Dataset Cleaning and Pre-processing:

4.4.1 Dataset Files:

The dataset came in two different files, Questions.csv and Tags.csv. The tags file had multiple rows for the same Question and required to be grouped by Question before combining the data in Questions.csv and Tags.csv to create a single dataset.

4.4.2 Text Data Cleaning workflow:

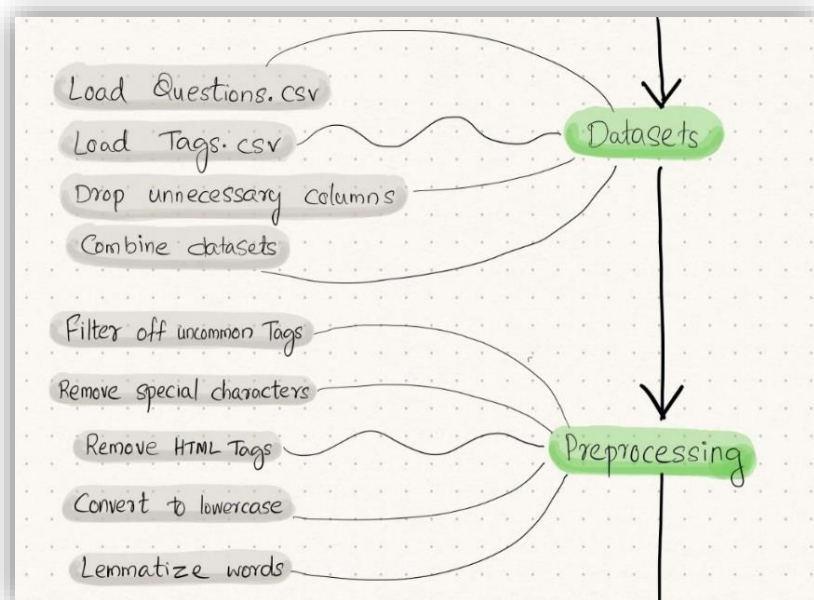


Fig5. Text Data Cleaning workflow

The title and text of in the dataset required to strip the html tags and then covert all the text to lowercase before using in the model. As a final step in the data cleanup, a process of lemmatization (Heidenreich, 2018) was followed to resolve the words to their dictionary form or root word (lemma). This allows to eliminate or replace the synonym words with root words and then use it for model creation. This allows the machine learning models to be more accurate since the model is working on the root forms instead of having to interpret synonyms of different words.

4.4.3 Steps for text cleaning:

The data must first be cleaned. It includes numerous html and url links, punctuation, single-letter alphabets, and stopwords, none of which provide much context for the subjects they refer to.

- There are columns in the dataset like creation_ID, creation_date and few more which will not make any progress and useful for the model building hence it was removed marked as unwanted columns.
- Once the required columns are framed, the data has to be passed through a filter checking for any missing, NULL, NaN and Infinite values in it. Because, the unsolicited values will make the model weak and to fail giving appropriate results. Using panda's library from python, the framed dataset is filtered out from those values.
- Now, the dataset is filtered and since it's the text data analysis, unwanted texts like, HTML tags, punctuations, alphabets through the regex and white spaces and stop words were removed for the model to understand the data better.
- Making the words lowercase is on such important process of text cleaning which is done with this data because, the NLP models would have a significant impact which will affect the final prediction to go wrong.
- Finally, lemmatization was used as the last step in the data cleanup to resolve the words to their dictionary form or root word (Heidenreich, 2018). By doing so, it is possible to remove or swap out synonyms for root words, which can then be used to create models. Since the machine learning models are working on the root forms rather than having to interpret synonyms of various words, the models are able to be more accurate as a result.

Hence, now the text data is cleaned by following the above steps. But, still the data needs to be prepared more before passing it into the models.

4.4.4 Steps for Feature Engineering of text data:

The data is cleaned by removing NULL, Nan values, HTML tags, stop words, making the text to lower case and lemmatizing the data. Now, the features need to be engineered before passing the data to the model building phase.

- **TF-IDF** - TF-IDF stands for term frequency inverse document frequency. Term Frequency summarizes how often a given word appears within a document and Inverse Document Frequency downscales words that appear a lot across documents. TF-IDF scores are the product of these two terms. Without inverse document term, words which appear more common (that carries little information) would be weighed higher and words appearing rare(which contains more information) would be weighed less.
- **Multi-label Binarization** - A binary vector is then used to represent each integer value, except for the integer's index, which is denoted by a 1 and all other values being zero. However, we are aware that any number of classes might be related to a multi-label classification problem. In MLB, the label (which can have multiple classes) is transformed into a binary vector such that all values are zero except the indexes associated for each class in that label, which is marked with a 1.
- **Tokenization** - Tokenization involves cutting the raw text into manageable pieces. Tokenization divides the original text into tokens, which are words and sentences. These tokens aid in context comprehension or model development for NLP. By examining the word order in the text, tokenization aids in interpreting the text's meaning.

- **Train and Test Split** – This is one of the significant phases after data cleaning and feature engineering processes. Once the data is engineered, the final framed data will be split into training data where the model will be trained and fitted using this data and testing data, where the fitted model will be evaluated and predicted using this data.

5. AI Models Implementation Phase:

5.1 Multi-class vs Multi-label classification problems:

There is a difference between multi-class classification problem and multi-label classification problem.

- **Multi-class classification** means a classification task with more than two classes; e.g., classify a set of images of fruits which may be oranges, apples, or pears. Multi-class classification makes the assumption that each sample is assigned to one and only one label: a fruit can be either an apple or a pear but not both at the same time.
- **Multi label classification** assigns to each sample a set of target labels. This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant for a document. A text might be about any of religion, politics, finance or education at the same time or none of these.

It is crucial to propose this multi-label classification problem as a binary classification for us to apply various ML models. This method is a multi-label-classification problem.

5.2 List of Multi-label classification ML models implemented:

Model No.	Model Name
1.	Linear Support Vector Classifier
2.	Stochastic Gradient Descent Classifier
3.	Logistic Regression
4.	Convolutional Neural Networks
5.	Bidirectional Encoder Representations from Transformers
6.	Long Short-Term Memory - RNN

5.3 Model Data Preparation:

Tags in this dataset is a categorical feature and since all ML algorithms need data in numerical form, it was needed to convert the 'Tags' column into categorical numerical values. But since the column contains multiple values a.k.a 'Tags', we cannot use any of the traditional categorical encoding like Label Encoder or One Hot Encoder or Pandas get_dummies method. The Multilabel Binarizer allows to encode more than one label at the same time. This is particularly useful when a categorical column has more than 1 value, like we have in our dataset.

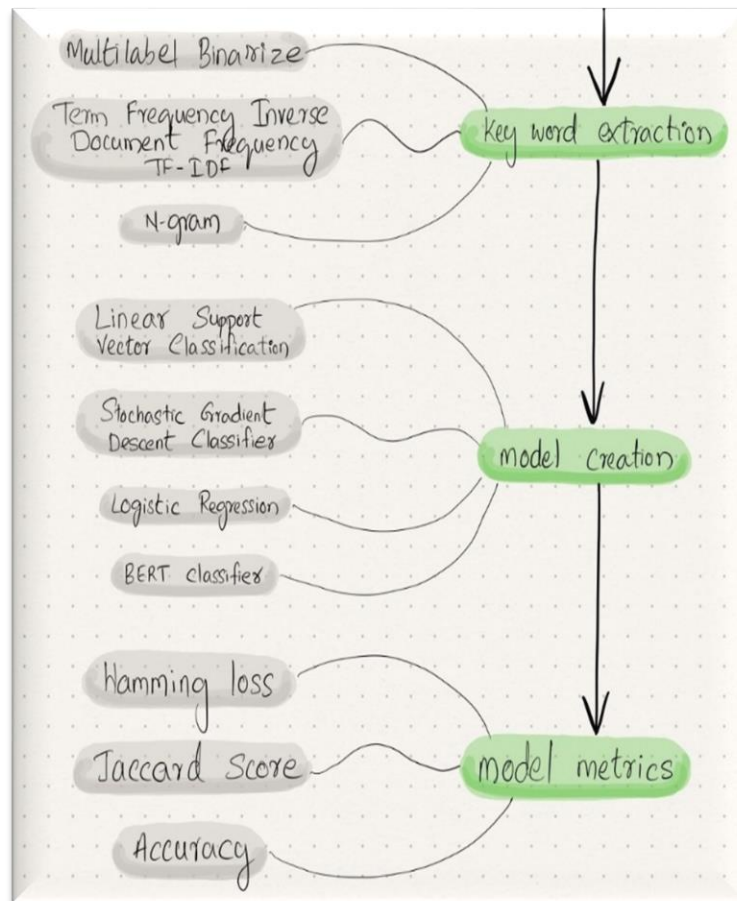


Fig6. Model Implementation and Evaluation

Similarly, the title and text of the content need to be converted to numerical form and this is done using Vectorization (Prabhu, 2019).

Out of the many ways to do vectorization like Count Vectorizer, Term Frequency Vectorizer or Hashing Vectorizer, the Term Frequency Inverse Document Frequency Vectorizer (TfidfVectorizer) was used for this project.

"TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents." (Prabhu, 2019).

After conversion of the dataset features in numerical and categorical arrays, the converted data can be used in the multi-label classifier models.

Multi-label classification is performed by breaking down the classification problem into multiple binary classification problems and training each binary classifier on each of the problems and then making prediction. This is done in our implementation using the `OneVsRestClassifier` which allows the use of `LinearSVC`, `SGDClassifier` or `LogisticRegression` which are binary classification models. We also tried other models like BERT, CNN, RNN and LSTM classifiers.

5.4 Model Evaluation:

Unlike traditional machine learning model accuracy, precision, recall and f1 score is not the right metrics for evaluating the performance of the trained models. The model performance needs to be evaluated differently for a multi-label classification model. Two metrics that was considered here are:

- Hamming Loss factor – “the fraction of the wrong labels to the total number of labels” (Nooney, 2018).
- Jaccard similarity (Uniqtech, 2020) – This is calculated by intersection of predicted tags and true tags over the union of the predicted and true tags.

The Linear SVC model was used for the implementation for this solution due to simplicity in deployment compared to other classifier models.

6. Establish Codebase Repository:

The Repurpost team did not provide any guideline about creating and sharing code with them, so NEU team Repurpost, created a GitLab repository to manage and maintain all the artifacts that will be delivered to the sponsor as soon as the project plan was finalized. GitLab provides integration with deployment tools and also provides web browser IDE to uploading the artifacts that includes Python code, Jupyter notebooks, User documentation, Web API models for deployment.

The folder structure of the GitLab repository is provided. The repository has been divided into 4 folders:

- `~/datasets/` – The “stack over flow” dataset that was used in this project.
- `~/notebooks/` – All the Jupyter python notebooks that was used for the different sections of this project
- `~/api/` – All the code that is supposed to be packaged and deployed as web API
- `~/documents/` – All the documents that include the architecture diagram, user flows and proposed wireframes.

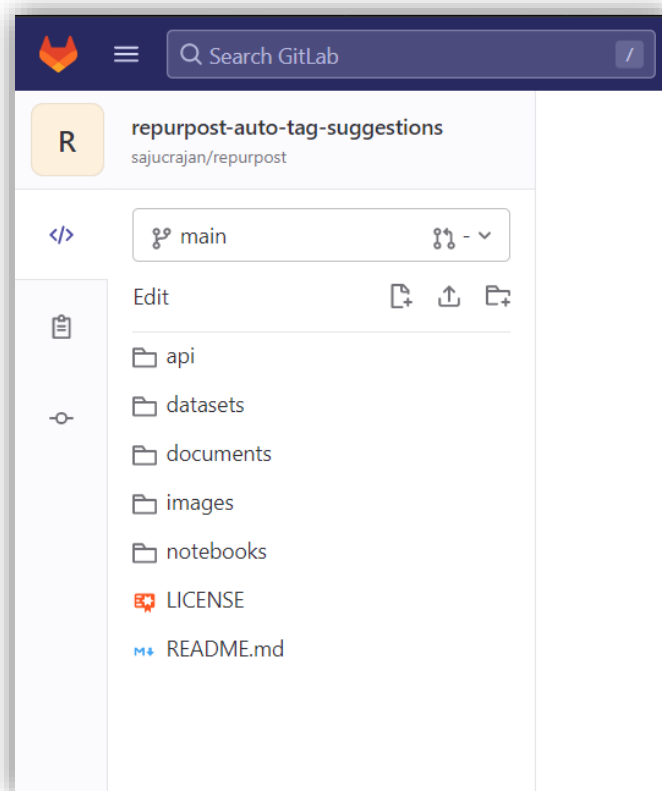


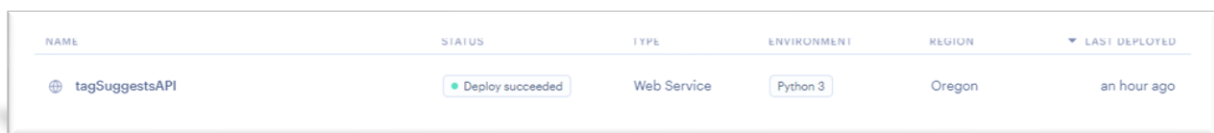
Fig7. Codebase Repository - Gitlab

7. Web API and Application Development:

7.1 Deploy web API using Flask:

- Once different models were created, tested and evaluated, it was decided that the Linear SVC model will be used for deployment as a web API.

- The standard process for model deployment using Flask (Vyas, 2019) involves exporting the trained model using the pickle package into the .pkl file and subsequently loading the .pkl file in the web API deployment.
- But this project needed the export of not only the model since the model expects text vectors as input and provided an array of binarized tags as output.
- This required the trained instances of MultiLabelBinarizer and TfidfVectorizer that can be used to transform the input text into array of vectors and inverse transform the model output to a list of tags. The Flask web API was tested in local.



NAME	STATUS	TYPE	ENVIRONMENT	REGION	LAST DEPLOYED
tagSuggestsAPI	Deploy succeeded	Web Service	Python 3	Oregon	an hour ago

Fig8. Web API Flask Deployment

For a professional delivery of the model, deploying or running the models in local machines was not the goal for our team. The team focused on trying to getting a cloud deployment that integrates with the GitLab repository and packages the exported model. Based on research, Render.com, vercel.com and Heroku.com was the options available. Render.com was the first deployment platform that we tried to deploy the web API and it satisfied all our requirements. Render.com automatically detect code changes in GitLab and retrigger deployment whenever there is a change in the model that is checked-in or the web API python code.

7.2 Develop Web interface and integrate web API:

The web interface or the web UI that is developed for this POC is a temporary interface to demonstrate the function of the web API and used to display the tags that will be suggested by the web API.

Keeping that under consideration, the team did not want to spend a large amount of time in UI development and was researching for options that will allow lean and rapid development and allow the team to focus on the web API integration.

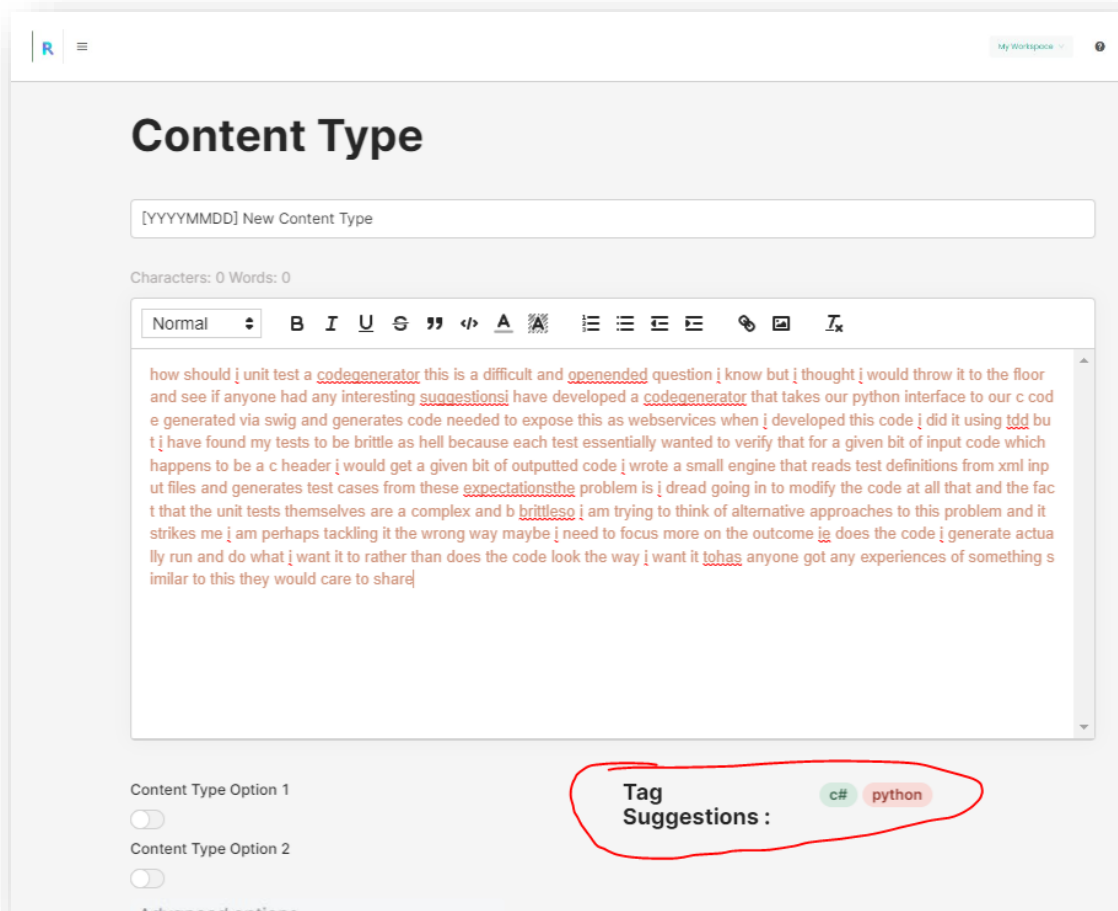


Fig9. Web application for tags generation system.

The above figure is the final deployment product of the web applications integrated using the flask web API by which the tags are generated based on the content given by the user which is successfully achieved by the AI model implementation.

- Retool.com was the perfect platform that enabled the team to create the required UI's and integrate with the web API.

- The team created two UI's and a resource integration with the web API that is deployed on Render.com.
- The web API is currently invoked in every keystroke in the text Area that provided the content text.
- This is not ideal but is a limitation of Retool.com and team is unable to provide a work around for that.

8. Discussion:

Elaboration of Model Results:

In this project, we created six tag prediction machine learning algorithms and compared their metrics to determine which works best for the web API implementation. The metrics we used to compare the models are the Hamming and Jaccard scores.

Hamming score: The Hamming score differs from the Accuracy score in a multi-label use case. Although the difference is minor, it has a significant impact on the metric, making the Hamming score a more viable metric for such tasks. The Hamming score's metric value interpretation is straightforward. More correct predictions result in a higher Hamming score. The greater the measured value, the greater the benefit. The best possible value is 1 (if the model correctly predicted all the outcomes), and the worst possible value is 0. (If a model failed to make even one correct prediction.)

Below is a bar graph that depicts the Hamming scores of the models:



Fig10. Model Results based on Hamming Score

Linear SVC got a score of 0.84, SGD Classifier got a score of 0.52, Logistic Regression got a score of 0.47, and CNN got a score of 0.47. The Linear SVC model has the highest Hamming score, which is 0.84. As previously stated, any hamming score close to 1 is considered good.

Jaccard Score: The Jaccard Index, also known as the Jaccard similarity coefficient, is a statistic used to determine how similar two sample sets are. The measurement emphasizes similarity between finite sample sets and is formally defined as the intersection size divided by the union size of the sample set. In this case, we need to estimate text similarity between tags and predicted tags in Natural Language Processing. In other words, Jaccard Score is defined as the intersection of two documents divided by their union, referring to the number of common words over a total number of words.

Below is a bar graph that depicts the Jaccard scores of the models:

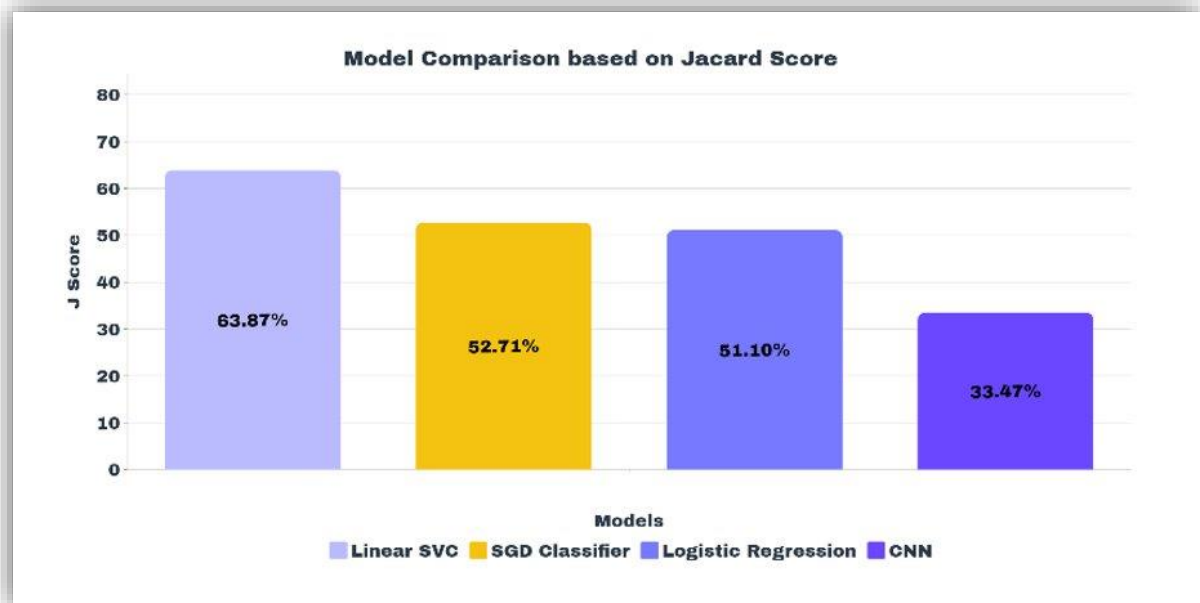


Fig10. Model Results based on Hamming Score

Linear SVC has a score of 63.87%, SGD Classifier has a score of 52.71%, Logistic Regression has a score of 51.10%, and CNN got a score of 33.47%. The Linear SVC model has the highest Hamming score, which is 63.87% which means that the Linear SVC model predicted tags output matches the most with the actual tags generated.

Looking at both these scores, we decided to go ahead with the Linear SVC model for the web API implementation as it obtained the highest scores.

Results of the Final Deployment:

This section displays the outcomes of the web applications that were integrated utilizing the AI-developed Flash web API. The key project objectives were addressed and developed in accordance with the specifications with the aid of the tools and technologies employed.

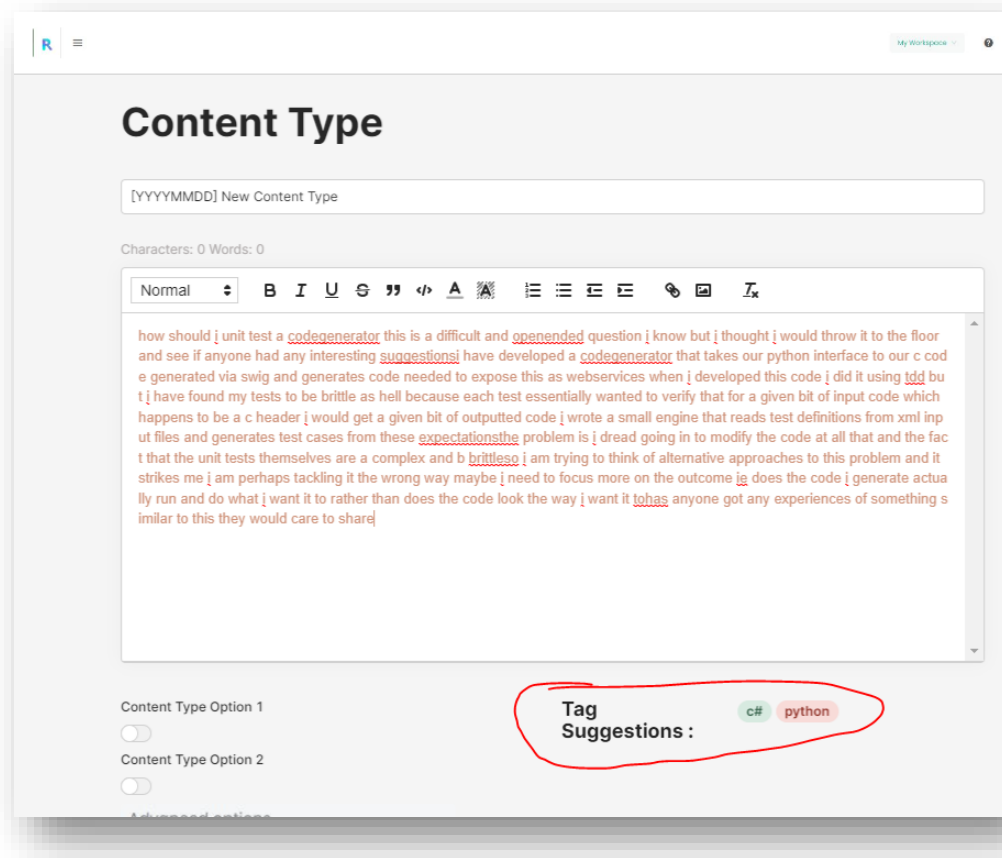


Fig11. Final Web UI for tag generation

The above figure is the final deployment product of the web applications integrated using the flask web API by which the tags are generated based on the content given by the user which is successfully achieved by the AI model implementation.

Further Improvements and Recommendations:**1) Improvements that could be made to predict more accurate tags:****A) Evaluate the model using appropriate metrics and train-validation-test the dataset.**

We should assess the model's precision, recall, and F1-score to determine whether changes to the model are beneficial. In a multilabel classification setting, the F1-score is typically used as the metric to optimize, but you should choose the metric based on the model's use case. For example, if the model assists humans in categorizing articles, recall may be more important than precision because there is always someone who can correct prediction errors.

B) Adjust each binary classifier's classification threshold. To optimize your metric, remember to tweak this parameter on the validation set.

C) Collect training data from various websites. In this tutorial, we scraped only Medium, but there are numerous other websites from which to collect data.

D) Taxonomies improve training data preparation. We data scientists tend to focus too much on the model and not enough on the data quality. Our training set consists of articles and tags assigned by their authors, but authors are frequently restricted in the number of tags that can be assigned. This means that the author must frequently choose which tags to assign among many plausible ones, based on some logic of tag visibility and specificity. As a result, our training set is contaminated because plausible tags are missing from the articles! To get around this issue, we could create a topic taxonomy and use it to clean up the dataset, assigning more generic tags whenever we see more specific tags assigned. For example, if an article has the tag Machine Learning, we can deduce that it also discusses Artificial Intelligence because AI is a hypernym of Machine Learning.

2) Improvements that could be made to the web application:

A – In our implementation of the project, we have separately given a feature for the user to check the tags relevant to what they are looking for. An upgrade in this area would be automatically displaying the predicted tags as the user starts typing their content so that they don't waste time in clicking and searching for tags separately.

B – Another upgrade would be to use a reinforcement learning technique wherein if a tag is found to be irrelevant to the topic, it sends negative feedback to the API to avoid displaying it again.

Conclusion

In this research, we provide a straightforward classifier algorithms that can predict tags for StackOverflow questions using just the question title and body. Future work should concentrate on improving the runtime of our model, as it is currently too sluggish to be used in reality, in addition to creating more complex features. This dissertation successfully combines the best model into the web API with the web application to implement the NLP idea utilizing several AI models to generate tags. This project can be designed and developed more effectively by using the suggested solutions that were addressed. This project produces tags based on the content of the user while adhering to the goals and specifications of Repurpost.

References

- Dask Dev. (n.d.). *Scale the Python tools you love*. Retrieved from Dask: <https://www.dask.org/>
- Deutschman, Z. (2019, December 3). *Multi-Label Text Classification*. Retrieved from Towards Data Science: <https://towardsdatascience.com/multi-label-text-classification-5c505fdedca8>.
- Jain, S. (2017, August 26). *Solving Multi-Label Classification problems (Case studies included)*. Retrieved from Analytics Vidhya: <https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/>
- Nooney, K. (2018, June 7). *Deep dive into multi-label classification..! (With detailed Case Study)*. Retrieved from Towards Data Science: <https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff>
- Prabhu, P. (2019, November 11). *Understanding NLP Word Embeddings — Text Vectorization*. Retrieved from Towards Data Science: <https://towardsdatascience.com/understanding-nlp-word-embeddings-text-vectorization-1a23744f7223>
- Proto.io. (2017, January 20). *Why Wireframes Are Important in the Design Process*. Retrieved from Medium: <https://protoio.medium.com/why-wireframes-are-important-in-the-design-process-de4e773e611>
- Stackoverflow. (2019, October 7). *StackSample: 10% of Stack Overflow Q&A*. Retrieved from kaggle: <https://www.kaggle.com/datasets/stackoverflow/stacksample>
- Team, R. (n.d.). *Build internal tools, remarkably fast*. Retrieved from Retool: <https://retool.com/>
- Uniqtech. (2020, August 20). *Understand Jaccard Index, Jaccard Similarity in Minutes*. Retrieved from Medium.com: <https://medium.com/data-science-bootcamp/understand-jaccard-index-jaccard-similarity-in-minutes-25a703fbf9d7>
- Vyas, H. (2019, November 30). *Deploy a machine learning model using flask*. Retrieved from Towards Data Science: <https://towardsdatascience.com/deploy-a-machine-learning-model-using-flask-da580f84e60c>

