DOCUMENT TAGGING SYSTEM FOR INFORMATION RETRIEVAL AND RELEVANCE

INTRODUCTION

Document tagging involves assigning meaningful labels or keywords to documents that represent their core content. This enables IR systems to:

- Enhance search precision: Tags help the system understand the context and content of documents, leading to more accurate retrieval results.

- Facilitate categorization: Tags allow documents to be grouped by themes or topics, making navigation and filtering easier.

- Support relevance ranking: Properly tagged documents help IR systems rank results based on their relevance to the query.

PROBLEM STATEMENT

- Inaccurate Tagging: Current document tagging systems often struggle to assign contextually accurate tags, especially for interdisciplinary or highly specialized papers.

- Incomplete or Irrelevant Tags: Many systems generate incomplete or irrelevant tags, making it difficult for users to retrieve the most relevant documents.

- Challenges in Discoverability: These inaccuracies hinder researchers from efficiently discovering relevant literature, impacting research quality and speed.

DATASET DESCRIPTION

- Using the NIPS Papers dataset, focused on machine learning and AI.

- High-quality abstracts ideal for extracting meaningful features.

Why NIPS Papers?

- They are highly specialized for the ML domain, aligning with tasks like topic modeling and keyword prediction.
- Abstracts are concise yet information-dense, ideal for extracting meaningful features.

MODEL ARCHITECTURE

1.Training Documents:
A collection of labeled research papers used for training the system.

2. Pre-processing:
Text cleaning and normalization (removal of stopwords, tokenization, stemming).

3. Keyword Extraction:
Identifies important key-phrases from the research papers.

4. Prediction Model:
A machine learning or deep learning model that predicts the most relevant domain.

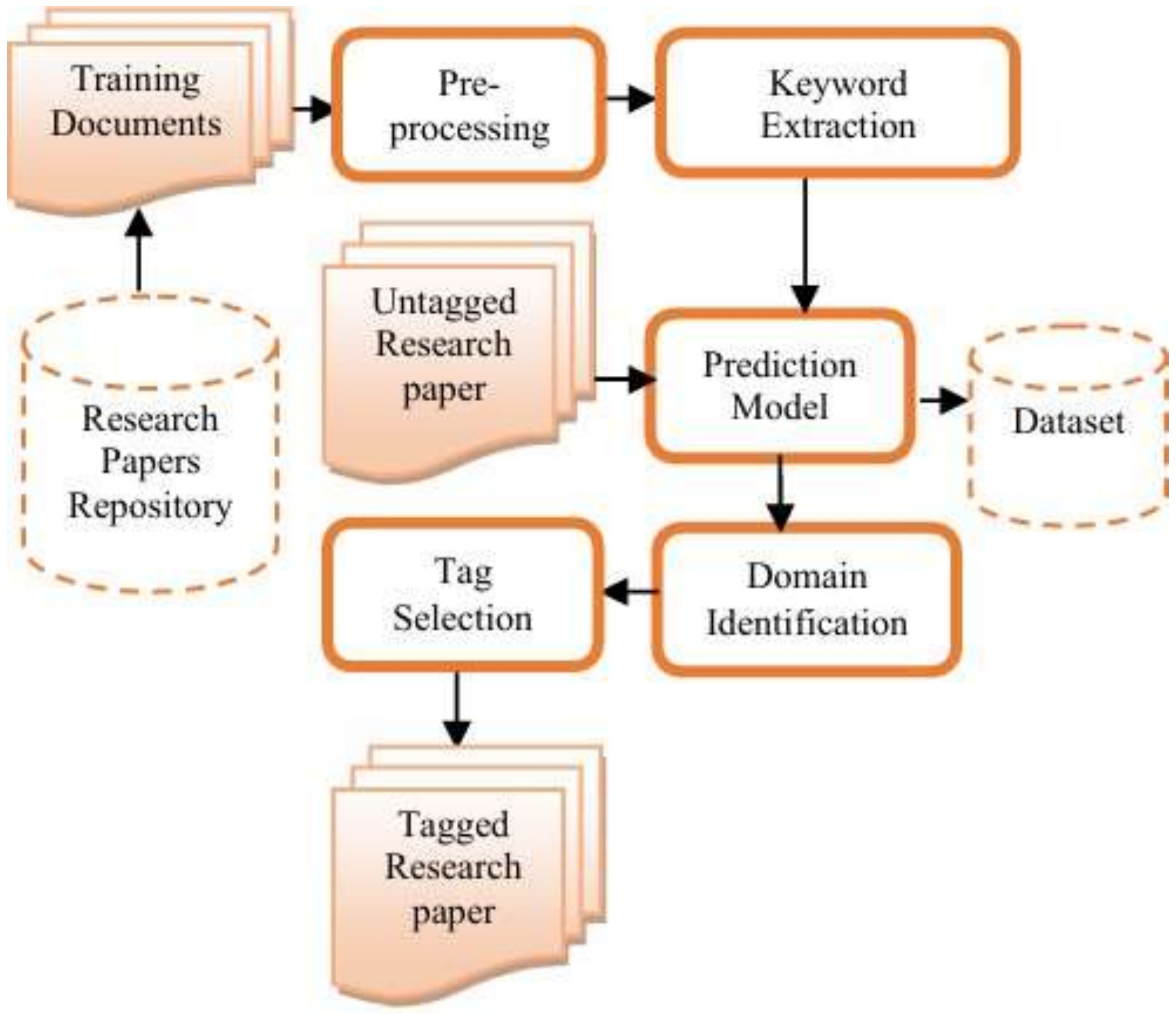

Fig. 1. System Architecture

MODEL ARCHITECTURE

5. Domain Identification:

Assigns a specific domain to each paper based on the prediction model output.

6. Dataset:

Serves as input for the model, consisting of previously labeled research data.

7. Tag Selection:

Selects appropriate tags (keywords and domains) for the untagged paper.

8. Tagged Research Paper:

The final output is a research paper enriched with selected tags and domain labels.

MODEL PREPARATION

1. Multi-Label Binarization:
    - Converts the extracted keywords into a multi-hot encoded format using MultiLabelBinarizer.
    - This creates a binary representation for each keyword.
2. Train-Test Split:
    - Splits the dataset into 80% training and 20% testing subsets to evaluate model performance effectively.
3. Removing Constant Labels:
    - Identifies and removes labels that are constant (always present), as they do not contribute to the model's learning.

MODEL PIPELINE

Classification

- Logistic Regression with One-vs-Rest (OvR):
  - A Logistic Regression classifier is trained with an OvR strategy, where one binary classifier is trained for each label.

Why Logistic Regression?

  - Computationally efficient.
  - Works well with high-dimensional data like BERT embeddings.
  - The OvR strategy handles multi-label classification effectively.
- Model Evaluation:
  - Predicted labels are compared to ground truth using accuracy and F1-score, assessing how well the model generalizes.

EVALUATION OF MODELS

- RAKE:
  - Precision: 50%
  - Recall: 33%
  - F1: 40%
  - RAKE struggles to align with the ground truth due to its reliance on statistical features alone.
- YAKE:
  - Precision: 66%
  - Recall: 50%
  - F1: 57%
  - Better performance due to context-aware extraction but still not optimal for complex language.

EVALUATION OF MODELS

- BERT:
  - Precision: 80%
  - Recall: 66%
  - F1: 72%
  - Achieves the best balance of precision and recall, making it the most reliable option for nuanced keyword extraction tasks.

KEYWORDS GENERATED BY EACH MODEL EXAMPLE

```
print("YAKE Keywords:", yake_keywords)
print("BERT Keywords:", bert_keywords)
```

```
Deep learning techniques have revolutionized natural language processing
RAKE Keywords: ['revolutionized natural language processing', 'deep learning techniques']
YAKE Keywords: ['Deep', 'processing', 'learning', 'techniques', 'revolutionized', 'natural', 'language']
BERT Keywords: ['learning techniques', 'natural language', 'deep learning', 'language processing']
```

- **"Deep learning" (BERT) is more precise because it captures a well-recognized concept without extra or missing words.**
- **"Deep" or "learning" (YAKE) are less relevant because they lack context.**
- **"Deep learning techniques" (RAKE) is less precise because it adds unnecessary details ("techniques") that dilute the focus.**

MODEL COMPARISON

Feature	RAKE	YAKE	BERT
Type	Statistical	Statistical	Deep Learning
Contextual Awareness	Low	Medium	High
Training Requirement	None	None	Pre-trained embeddings
Speed	Fast	Moderate	Slow
Accuracy	Basic keywords	Moderately accurate	Highly accurate
Scalability	Works for small datasets	Scalable	Requires significant resources
Evaluation Metrics (Precision, Recall, F1)	Lower (0.5, 0.33, 0.4)	Moderate (0.66, 0.5, 0.57)	Higher (0.8, 0.66, 0.72)

CONCLUSION

1. For Scalability: Use YAKE, as it offers a balance between accuracy and computational efficiency.
2. For Precision and Depth: Use BERT for tasks requiring high-quality keyword extraction, especially in academic or complex textual datasets.
3. For Simplicity: Choose RAKE for straightforward and small-scale keyword extraction needs.

Thank You