# Section 4: System Architecture and Algorithms Used

## 4.1 Architecture Overview

The system pipeline designed for the Make Data Count (MDC) citation classification task follows a structured process composed of six major components: raw XML extraction, text preprocessing, vectorization, and classification using both machine learning and deep learning approaches. Figure 5 visualizes this architecture.

## 4.2 Preprocessing Pipeline

Each article’s full text is stored in XML format. The pipeline starts by:  
- Parsing XML trees using Python’s xml.etree.ElementTree  
- Locating dataset mentions via regex patterns (e.g., DOI format: 10.1234/abcd)  
- Extracting the surrounding sentence (context) to be labeled as Primary, Secondary, or Missing  
  
After extraction, the data is cleaned by removing rows with:  
- Empty fields or “Missing” labels  
- Contexts shorter than 5 tokens (to exclude noise)  
  
This ensures that the final dataset (train\_data\_cleaned) consists of meaningful textual samples ready for modeling.

## 4.3 Tokenization and Feature Engineering

Two distinct tokenization strategies were used:

A. TF-IDF Vectorization:  
- Transforms each sentence into a numerical vector based on term frequency-inverse document frequency  
- Captures surface-level lexical patterns  
- Used in conjunction with classical ML models  
Implemented via: TfidfVectorizer(stop\_words='english', ngram\_range=(1,2))

B. Transformer-Based Embeddings (Planned/Future Work):  
- Models like BERT or RoBERTa tokenize using WordPiece encoding  
- Sentences are represented in dense, context-aware vector form  
- Ideal for capturing semantic differences between “used data” (Primary) vs “based on data” (Secondary)

## 4.4 Machine Learning Algorithms

Several models were evaluated using TF-IDF vectors:

Logistic Regression (Baseline):  
- Simple linear model used for multiclass classification  
- Benefits from interpretability and fast training  
- class\_weight='balanced' was used to counter dataset imbalance

Support Vector Machine (SVM) + TF-IDF:  
- Effective for high-dimensional sparse text  
- Uses hyperplanes to separate classes  
- Can be adapted to non-linear kernels for complex boundaries

Random Forest with Bag-of-Words (BoW):  
- Ensemble model that builds multiple decision trees on bootstrapped BoW features  
- Reduces overfitting compared to single decision trees  
- Performs moderately well but lacks semantic understanding

## 4.5 Deep Learning Models

Planned experimentation includes:

BERT Fine-Tuning:  
- BERT (Devlin et al., 2019) is pre-trained on masked language modeling  
- Fine-tuning adjusts BERT weights on citation classification task  
- Outperforms TF-IDF + ML in most academic benchmarks

BiLSTM (Bidirectional Long Short-Term Memory):  
- Captures context from both directions in a sequence  
- Effective when sentence length and order matter  
- Can be stacked on top of pretrained word embeddings (e.g., GloVe, Word2Vec)

## 4.6 Architecture Diagram

Figure 5 shows the full pipeline—from XML ingestion to both machine learning and deep learning outputs. The modular design allows easy switching between model types and supports ensemble methods.

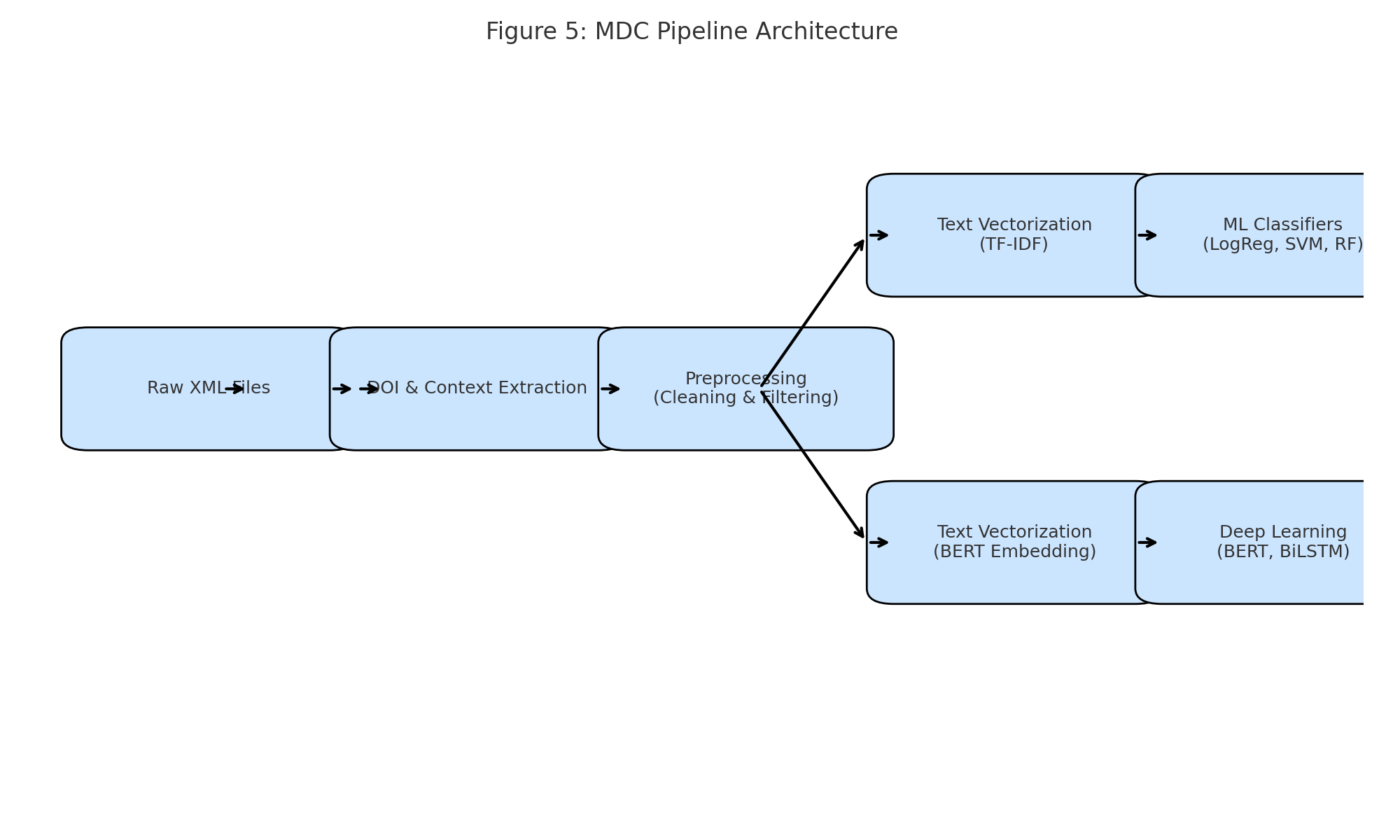


Figure 5: MDC Architecture Pipeline

## References

Devlin, J., Chang, M. W., Lee, K. and Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint. Available at: https://arxiv.org/abs/1810.04805 [Accessed 19 June 2025].

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M. et al., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, pp.2825–2830.