Polysemy in Natural Language Processing

Mohamed Ahmed Mansour

July 28, 2025

Text Preprocessing

- **Text Cleaning** — Text cleaning involves removing unnecessary characters and formatting issues, such as punctuation, numbers, or extra spaces. This step improves the quality of the input data for downstream tasks like tokenization or vectorization.

- **Tokenization** — Tokenization is the process of splitting text into smaller units like words or subwords. It is crucial for converting unstructured text into analyzable units. Common types of tokenization include:
  - **Sentence Tokenization**: Splits a document into individual sentences. Useful for tasks like summarization or machine translation.
  - **Word Tokenization**: Splits sentences into individual words or tokens. This is the most common form of tokenization used in NLP pipelines.
  - **Character Tokenization**: Splits text into individual characters. Used in tasks like character-level language modeling or when dealing with unknown words.

- **Stopword Removal** — Stopwords are common words like “and”, “the”, and “is” that do not contribute significant meaning. Removing them helps focus the model on more meaningful content.

- **Stemming** — Stemming is the process of reducing a word to its root or base form by removing suffixes. It uses rule-based methods without considering the context or the actual meaning of the word. For example, “playing”, “played”, and “player” may all be reduced to “play”. Stemming is faster but often less accurate than lemmatization, which uses vocabulary and morphological analysis.

- **Lemmatization** — Lemmatization reduces words to their base or dictionary form (e.g., “running” to “run”) using linguistic analysis. It is preferred over stemming for better accuracy.

- **Comparison between Stemming and Lemmatization**

Aspect	**Stemming**	**Lemmatization**
Definition	Removes suffixes to reach the root form of a word using heuristics	Reduces a word to its base or dictionary form using linguistic rules
Method	Rule-based (often using crude chopping)	Vocabulary-based + morphological analysis
Accuracy	Lower; can produce non-real words	Higher; returns actual valid words
Example	“studying”, “studies” → “studi”	“studying”, “studies” → “study”
Speed	Faster	Slower
Use Cases	When speed is more important than precision	When grammatical correctness and context matter

Table 1: Comparison between Stemming and Lemmatization

- **POS Tagging** — Part-of-speech tagging assigns a grammatical category to each token (e.g., noun, verb). It is used in advanced NLP tasks like parsing and named entity recognition.

Word Representation

- **Encoding**
  - – **One-Hot Encoding** — Transforms categorical variables into binary vectors. Each word is represented as a vector with one ‘1‘ at the index of that word and ‘0‘s elsewhere.
  - – **Label Encoding** — Converts categorical labels (words) into integer values. Each word gets assigned a unique numeric ID. It’s useful in preparing categorical data for machine learning models.

Aspect	**One-Hot Encoding**	**Label Encoding**
Definition	Represents words as binary vectors with a single high bit	Assigns each word a unique integer value
Output Format	Binary matrix	Integer vector
Interpretability	Easy to interpret	Less intuitive for categorical comparisons
Sparsity	High (mostly zeros)	Low
Use Cases	Suitable for non-ordinal categories	Suitable for tree-based models

Table 2: Comparison between One-Hot and Label Encoding

- **Bag of Words (BoW)** — Represents text as a vector of word occurrence counts, disregarding grammar and word order but keeping multiplicity.
- **TF-IDF (Term Frequency-Inverse Document Frequency)** — TF-IDF adjusts the word counts in BoW based on how frequently words appear across documents. It reduces the importance of common words.
- **N-Grams** — N-grams are contiguous sequences of n words (e.g., unigrams = 1, bigrams = 2). They capture local word context in the text.
- **Occurrence Matrix** — A matrix showing how many times each word appears in each document. Each row is a document; each column is a vocabulary term.
- **Co-occurrence Matrix** — Measures how often words appear together in the same context (e.g., same document). Useful for understanding semantic relationships between words.

Embedding

Embedding Techniques

- **Word Embedding:** A method to represent words in dense vector space, capturing semantic meaning and relationships.
- **Word2Vec:** Predictive embedding model by Google using Skip-Gram or CBOW architectures to learn word associations from large corpora.
- **GloVe:** (Global Vectors for Word Representation) by Stanford. It combines global matrix factorization and local context windowing to generate word vectors.
- **FastText:** Developed by Facebook, it improves on Word2Vec by considering subword information, allowing better handling of rare and out-of-vocabulary words.
- **ELMo:** (Embeddings from Language Models) Provides context-aware word representations using deep bidirectional LSTMs trained on a language modeling task.

- **BERT:** (Bidirectional Encoder Representations from Transformers) A transformer-based model that provides contextualized embeddings using masked language modeling.

Comparison: Word2Vec vs GloVe

Feature	**Word2Vec**	**GloVe**
Training Method	Predictive (learns by predicting context words)	Count-based (factorizes word co-occurrence matrix)
Context Window	Local	Global
Computational Efficiency	Faster on large corpora	Requires more preprocessing (matrix factorization)
Accuracy	High	Comparable or slightly better for semantic tasks

Dual-Branch Fake News Detection Framework

System Overview

- **Goal:** To classify whether a news article is real or fake by using two sources of information:
  - **Text Branch (BERT):** Understands the semantics and context of the article.
  - **Knowledge Branch (TransE):** Matches extracted factual information against a structured knowledge graph.

- **Main Components:**
  - **Triplet Extraction:** Using REBEL to generate (Head, Relation, Tail) triplets from text.
  - **Fuzzy Matching:** Aligns extracted triplets to entries in the knowledge graph using Levenshtein distance.
  - **Dual Branches:** BERT encodes text semantics; TransE scores triplet logic.
  - **Interaction Module:** Fuses scores from both branches to output the final prediction.

Triplet Extraction

- Uses **REBEL (Relation Extraction By End-to-end Language generation)** based on BART to extract structured triplets.
- Example: “Einstein invented the internet” $\rightarrow$ (Einstein, invented, internet)

Fuzzy Matching with Knowledge Graph

- Purpose: Align triplets to a large external knowledge graph (e.g., CSKG).

- Uses **Levenshtein Distance** to compute similarity between triplet components and knowledge base entries.

Text Branch: BERT-Based Semantic Encoder

- Tokenizes and encodes the article using **BERT**.
- Produces contextual embeddings that represent the overall meaning of the document.
- Output: `V_D` (semantic vector from document).

Knowledge Branch: TransE Triplet Embedding

- **Triplet Aggregation Module:**
  - Uses **TransE** to embed each triplet as a vector.
  - Combines all triplet embeddings using **MLP-Mixer** to form a document-level knowledge vector Y_D.
- **Triplet Scoring Module:**
  - Computes logical score of each triplet: $\text{score}(h, r, t) = \|h + r - t\|$
  - Final rationality score p_{hrt} is computed based on all triplet scores.

Interaction Module and Final Prediction

- Combines outputs from both branches:

$$p_{predict} = g \cdot p_{text} + (1 - g) \cdot p_{hrt}$$

- g is a learnable weight parameter to balance text and knowledge importance.
- If $p_{predict} >$ threshold, then the article is classified as **Fake News**.

Loss Function

- The model is trained using Binary Cross-Entropy Loss.
- L2 Regularization is added to prevent overfitting.

Comparison Table: Text Branch vs Knowledge Branch

Aspect	**Text Branch (BERT)**	**Knowledge Branch (TransE)**
Purpose	Understands article meaning using context	Verifies logical truth via factual reasoning
Input	Full text article	Extracted triplets (h, r, t)
Method	Transformer-based encoding (BERT)	Embedding + scoring using TransE
Strength	Captures semantics, tone, and grammar	Incorporates real-world knowledge and logic
Output	Semantic vector (V_D)	Rationality score (p_hrt)

Table 3: Comparison between Text Branch and Knowledge Branch