Knowledge Graph Construction and Embedding
Technical Report

Web Datamining and Semantics
MESIIN480224

HAMADEH Rayan
AFFES Nour

Table of Contents

1. Project Overview and Methodology

This project involved building a complete knowledge graph pipeline from web-scraped news articles. The goal was to transform unstructured text into a structured knowledge representation that captures entities and their relationships, then learn embeddings to enable semantic queries and predictions.

Our methodology followed a five-stage pipeline:

1. Web scraping of CNN technology articles
2. Text preprocessing and cleaning
3. Named entity recognition and relation extraction
4. Knowledge graph construction
5. Graph embedding and enrichment with external knowledge

Technologies used included Python, BeautifulSoup, spaCy, sklearn-crfsuite, RDFLib, and PyKEEN.

2. Implementation Details

2.1 Web Scraping

We implemented web scraping using Python's requests and BeautifulSoup libraries. Initially, we attempted to scrape Reuters but encountered anti-scraping measures. We pivoted to CNN's technology section, where we successfully extracted 10 articles.

Our scraper handled:

- Article title extraction
- Content extraction with proper paragraph structure
- Publication date retrieval
- URL recording for reference

2.2 Text Preprocessing

We developed a robust text preprocessing pipeline including:

- HTML tag removal
- Lowercase conversion

- Punctuation handling (preserving hyphens for compound words)
- Stop word removal
- Tokenization
- Lemmatization

This cleaned text served as input for our entity and relation extraction components.

2.3 Named Entity Recognition

We implemented and compared two NER approaches:

1. **spaCy's pre-trained model**: We utilized the en_core_web_sm model which provided out-of-the-box performance.
2. **Conditional Random Field (CRF)**: We attempted to train a custom CRF model using the CoNLL-2003 dataset.

The spaCy model significantly outperformed our CRF implementation, extracting approximately 949 entities from the 10 articles, with the following distribution:

- PERSON: ~120 entities
- ORGANIZATION: ~150 entities
- LOCATION: ~95 entities
- MISC/DATE/TIME/OTHER: ~140 entities

2.4 Relation Extraction

We implemented relation extraction using spaCy's dependency parsing. The initial implementation focused on basic subject-verb-object patterns but yielded limited results. We enhanced it with:

Extended pattern recognition:

- Subject-verb-object patterns
- Entity-verb-prep-Entity patterns
- Adjacent entity relationships

Custom rules for specific relationship types:

- Company-location relationships (headquartered in)
- Person-organization relationships (works for, founded)

- Product-company relationships (manufactured by)

2.5 Knowledge Graph Construction

We used RDFLib to construct the knowledge graph, converting extracted entities and relations into RDF triples. Our namespace structure included:

The resulting graph contained:

- Nodes: 364 entities
- Edges: 554 relationships
- Triple patterns: Subject-Predicate-Object format

2.6 Entity Linking and Augmentation

We implemented entity linking to DBpedia to enrich our knowledge graph.

This augmentation significantly expanded our graph:

- Added ~15 new entities from DBpedia
- Incorporated ~30 new relationships
- Provided external identifiers for entities

2.7 Knowledge Graph Embedding

We trained three embedding models using PyKEEN:

1. **TransE**: Represents relationships as translations in the embedding space
2. **DistMult**: Uses a bilinear model with diagonal matrices for relations
3. **ComplEx**: Leverages complex-valued embeddings for asymmetric relations

Due to PyKEEN version compatibility issues (v1.11.0), we had to adapt our implementation approach, working directly with the model state dictionary and tensor operations.

Each entity was embedded as a 50-dimensional vector, capturing semantic relationships in the graph.

3. Results and Evaluation

3.1 Entity Extraction Performance

spaCy successfully identified 505 entities across the news articles. The most frequently mentioned locations included:

- California: 2 mentions
- Shenzhen: 2 mentions
- New York: 2 mentions

3.2 Knowledge Graph Quality

Our graph consisted of 364 entities and 554 meaningful triples, exceeding the minimum requirement of 50 triples. All triples used proper RDF syntax and appropriate namespaces.

3.3 Embedding Analysis

Embeddings analysis revealed semantic clusters of related entities. For example:

- Tech companies clustered together (Google, Microsoft, Apple)
- Locations formed distinct clusters
- People were grouped by their professional domains

Entity similarity calculation between entities showed meaningful relationships:

- Entity ID 0 - Entity ID 1: 0.1636 (moderate similarity)
- Entity ID 2 - Entity ID 3: 0.1665 (moderate similarity)

3.4 Augmentation Impact

Data augmentation from DBpedia showed measurable improvements:

- Entity embeddings changed after augmentation, with similarity scores between original and augmented versions ranging from 0.2548 to 0.9293
- The additional context improved semantic relationships
- Visualizations showed clear integration of new knowledge

4. Challenges and Solutions

4.1 Web Scraping Challenges

Challenge: Anti-scraping measures on Reuters. **Solution**: Switched to CNN with appropriate headers and rate limiting.

4.2 NER Limitations

Challenge: CRF model training complexities with CoNLL dataset. **Solution**: Leveraged spaCy's pre-trained model for better out-of-the-box performance.

4.3 Relation Extraction Difficulties

Challenge: Basic dependency parsing missed many implicit relationships. **Solution**: Implemented multiple pattern recognition approaches and custom rules.

4.4 PyKEEN Version Compatibility

Challenge: PyKEEN 1.11.0 had different API structure than current documentation. **Solution**: Developed version-specific approaches to access embeddings and perform computations.

5. Visualization and Analysis

Visualizations revealed interesting patterns in our knowledge graph:

- t-SNE plots showed clear entity clustering by type
- The augmented knowledge graph visualization demonstrated integration of DBpedia entities (shown in red) with original entities (shown in blue)
- Embedding similarity analysis highlighted semantic relationships not explicitly stated in the text

6. Conclusion

This project successfully demonstrated the entire pipeline from unstructured news text to a queryable knowledge graph with semantic embeddings. The combination of NLP techniques, graph representation, and neural embeddings created a powerful framework for knowledge extraction and representation.

The most successful aspects were:

1. Entity extraction with spaCy
2. Knowledge graph construction with RDFLib
3. Embedding generation despite API challenges
4. DBpedia integration for knowledge enrichment

The resulting knowledge graph captures meaningful relationships between entities in the technology domain, providing a foundation for semantic search, recommendation systems, or further knowledge discovery applications.