Automatic Knowledge Graph Construction from Unstructured Text

Rhythm Muhtadyuzzaman Syed
Georgia Tech Research Institute
msyed32@gatech.edu

Zach Welz
Georgia Tech Research Institute
Zach.Welz@gtri.gatech.edu

Santiago Balestrini
Georgia Tech Research Institute
Santiago.Balestrini@gtri.gatech.edu

Abstract

Knowledge Graphs are valuable data models useful for many Natural Language and Artificial Intelligent systems. A challenge with generating knowledge graphs is that they are time-intensive and expensive to build manually. In this work, we propose the framework for automatically constructing and augmenting knowledge graphs from unstructured text corpora with minimal human supervision. We evaluate our findings on an existing ontology domain including an Entity Linking task for graph expansion. The source code is available here[1].

1 Introduction

Knowledge graphs are readily used in many industrial systems today due to their flexibility in knowledge representation and power in reasoning. Data points are represented as structured knowledge such as a set of triples in a graph database which allows for more expressiveness in predicate and rule-based logic and increased query speed for real-time results. A knowledge graph is extremely useful for organizing real-life data as it utilizes ontologies as the framework to fill in real data about a domain and can scale up to large interconnections of various classes. An example use case is Google Search which transitioned from being a simple information engine to a knowledge engine by utilizing graphs. Given a search query, Google's knowledge graph understands how that search is related to other topics and returns results of most relevance. Using the power of data federation with graphs, Google also displays information and related topics in a UI panel by aggregating data from distributed sources.

In this work, we focus on the problem of developing a knowledge extraction pipeline that automatically is able to generate a knowledge graph given a set of documents containing unstructured text. We use machine learning and natural language processing methods to produce and augment RDF data for applications in business intelligence. The goal is to develop an application that offers full automation with limited human supervision ultimately speeding up business processes and time to production. We also allow the option of keeping a human in the loop, hence supporting more complex workflows. There is a direct need to develop the capability of automatic semantic data processing and model creation on unstructured text documents for the purposes of creating more powerful knowledge representation.

We aim to answer the following question in this work: Given a set of documents of **Unstructured Text**, can we develop a pipeline for building a **Structured Semantic Data Model** in the form of a **Knowledge Graph** for integration with **Downstream Applications**?

2 Related Work

In the work by Wu et al. (2019), the authors discuss the approach of teams for the Knowledge Graph construction contest at ICDM/ICBK 2019. The methods focused on entity recognition, relation extraction, and coreference resolution using both original and existing systems. The winning team UWA used spaCy for POS tagging, a Bi-LSTM deep learning model for relation classification, and NeuralCoref by Huggingface for coreference resolution. The work also discusses key techniques in knowledge graph construction by discussing rule-based, machine learning, and deep learning-based approaches offering a wide variety of options for developing a pipeline.

In the work by Yu et al. (2020), the authors constructs knowledge graphs in 3 steps: conversion, encoding, and surface-entity linking. In the conversion step, they utilize OpenIE from AllenAI on every sentence to generate a list of entity-relation

[1] https://github.com/jupyrdf/ipykgb

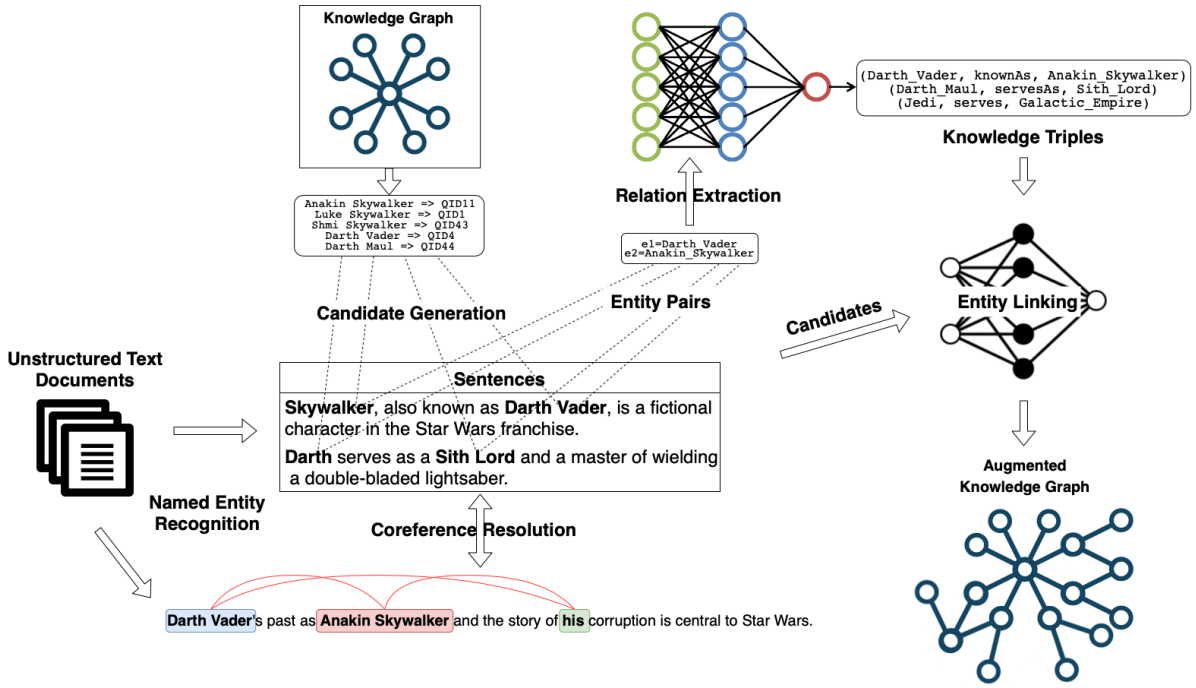

Figure 1: Knowledge Extraction Toolkit (KET) Workflow Visualization

triples. In the encoding step, they utilize BERT embeddings to encode each triple to form contextualized information. For example, the work 'Apple' contains a different meaning in the context in which it is used. In the surface-entity linking step, the authors creates a graph structure from the extracted triples using a cosine distance between the encodings to determine the point of insertion in the existing graph.

Ji et al. (2021) provides a survey and comprehensive review of knowledge graph related topics covering graph representation learning, acquisition and completion, temporal KGs, and applications. Focusing on knowledge acquisition, the survey divides this task into three categories: completion, relation extraction, and entity discovery. Completion involves expanding existing graphs using methods such as embedding-based ranking, relation path reasoning, rule-based reasoning, and meta relational learning. Relation extraction utilizes neural methods with attention mechanism, graph convolutional networks, adversarial training, reinforcement learning, deep residual learning, and transfer learning. Entity discovery describes finding terms using recognition, disambiguation, and alignment.

3 Method

The methodology in this work is a four step process as seen in Figure 2. Given a set of unstructured text documents, we pre-process the text to eliminate noise such as page breaks and headers and identify keywords with named entity recognition. Next we perform coreference resolution to disambiguate word references in the collection of sentences. In the next step, we perform relation extraction between the discovered terms to propose a set of new triples for augmenting the existing knowledge graph. In the final step, we perform entity linking that trains a model to disambiguate terms and IDs and determine the ideal insertion point for the new triples into the existing graph. This offers a more complete knowledge graph that can be used in a variety of applications.

3.1 Dataset Creation and Preprocessing

To evaluate the knowledge graph construction pipeline, we develop an automatic dataset creation system that collects unstructured text from auxiliary data sources from the web such as Wikipedia. In this example, we focus on an existing ontology domain that offers a wide variety of entities and complexity. For this purpose, we develop a corpus of Wikipedia articles focusing on the domain of the Star Wars fictional universe. By providing a start topic such as 'Star Wars' and the total number of desired topics such as '100', the system performs a multi-hop traversal to each related topic, resulting in a collection of articles spanning details about

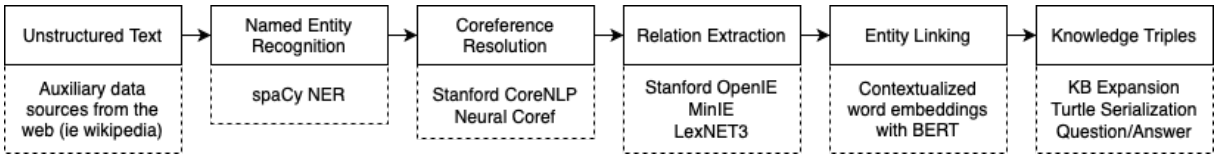

Figure 2: KET Tools and Systems Flowchart

Star Wars specific characters, books, movies, vehicles, and weapons. For the preprocessing step, we use Named Entity Recognition (NER) and Coreference Resolution systems which we detail in the following sections. We also use an NP-chunker to search for individual noun phrases in each sentence in the corpus. The preprocessing step eliminates noise in the original corpus such as section headers, line breaks, word ambiguity, and pronoun references.

3.2 Term Extraction

For term extraction, we utilize NER systems to locate and classify entities in text. This is used to identity people based on their names, locations, time, numerical values, etc. We evaluate two methods of entity extraction involving spaCy[2] and StanfordNER[3]. SpaCy offers a neural based NER system using subword features, bloom embeddings and residual CNNs. StanfordNER offers a trained linear chain Conditional Random Field (CRF) sequence model. Given a large body of text, spaCy NER performs the best, being able to recognize location, work of art, and dates of varying structures as compared to StanfordNER with both NLTK and spaCy tokenizers.

3.3 Coreference Resolution

Coreference resolution is a task in NLP that analyzes the references in text that point to the same entity. Examples of this include resolving the entity that a pronoun references in a span of multiple sentences. For example, given a sentence such as 'My brother loves to play basketball. He plays as a point guard', the resolved text will be 'My brother loves to play basketball. My brother plays as a point guard'. We evaluate two coreference systems: Stanford CoreNLP[4] and Neural Coref[5]. Stanford CoreNLP provides an NLP library derived from linguistic techniques for tasks such as POS tagging, dependency and constituency parsing, and coreference resolution. Neural Coref is an extension to spaCy that utilizes a model based prediction pipeline to resolve coreference clusters in a sentence.

3.4 Relation Extraction

Relation extraction (RE) is used to parse relationships between two entities within a sentence. This is helpful for knowledge graphs that are built from triple relations of (entity1, relation, entity2). We evaluate the performance of RE on the following systems: MinIE[6], Stanford OpenIE[7], Hearst Patterns[8], and OpenNRE[9] pre-trained models. The comparisons of the challenges and opportunities for each system can be seen in Table 2. MinIE and OpenIE are systems that focus more on the linguistic structure of text for relation extraction by looking at the dependency paths between entities. OpenNRE is a neural network based system that loads pre-trained models and predicts a relation given a sentence and the spans of two entities in the sentence.

To perform relation extraction, we leverage Stanford OpenIE to extract triples from the sentences in the cleaned dataset. Stanford OpenIE focuses on finding new relations from text with a part of speech tagger and dependency parser to extract paths. A big benefit of using an information extraction system such as this is that we do not have to define the desired relations beforehand. As compared to training a neural model which requires predefined relation classes, Stanford OpenIE discovers new relations. For every sentence in the cleaned dataset, we create a collection of relations extracted by the Stanford OpenIE system. For example, given a sentence: 'the-character is a part of the-game', the desired relation will be 'a-part-of', resulting in the triple: ('the-character', 'a-part-of', 'the-game'). A big benefit of the Stanford OpenIE system is that it is able to extract more triples in fine-grain detail as compared to other OpenIE sys-

[2]https://spacy.io
[3]https://nlp.stanford.edu/software/CRF-NER.html
[4]https://stanfordnlp.github.io/CoreNLP/
[5]https://github.com/huggingface/neuralcoref
[6]https://github.com/uma-pi1/minie
[7]https://nlp.stanford.edu/software/openie.html
[8]https://github.com/mmichelsonIF/hearst$_patterns_python$
[9]https://github.com/thunlp/OpenNRE

```
*** Creating Dataset ***
Topic0: Star Wars: Episode II - Attack of the Clones, Sentences: 341, Total: 341
Topic1: Ahsoka Tano, Sentences: 226, Total: 567
Topic2: List of natural satellites, Sentences: 95, Total: 662
Topic3: Lego Star Wars: The Complete Saga, Sentences: 88, Total: 750
Topic4: Daisy Ridley, Sentences: 99, Total: 849
Topic5: The Empire Strikes Back (novel), Sentences: 50, Total: 899
Topic6: Star Wars Jedi: Fallen Order, Sentences: 144, Total: 1043
Topic7: Changes in Star Wars re-releases, Sentences: 296, Total: 1339
Topic8: The Mandalorian, Sentences: 305, Total: 1644
Topic9: Star Wars prequel trilogy, Sentences: 137, Total: 1781
Topic10: Star Wars sequel trilogy, Sentences: 435, Total: 2216
Topic11: Lego Star Wars: The Video Game, Sentences: 146, Total: 2362
```

Figure 3: Topic generation example for Star Wars

```
Entities in the KB: ['https://swapi.co/resource/human/67', 'https://swapi.co/resource/human/35', 'https://swapi.co/resource/planet/8', 'https://swapi.co/resource/human/11', 'https://swapi.co/resource/toydarian/40', 'https://swapi.co/resource/human/51', 'https://swapi.co/resource/planet/9', 'https://swapi.co/resource/yodasspecies/20', 'https://swapi.co/resource/human/22', 'https://swapi.co/resource/planet/1']
Aliases in the KB: ['AnakinSkywalker', 'Tatooine', 'Amidala', 'Watto', 'Fett', 'Anakin', 'Dooku', 'MaceWindu', 'Mace', 'Windu', 'PadméAmidala', 'Coruscant', 'Skywalker', 'Yoda', 'Naboo', 'BobaFett', 'Padmé', 'Boba']
Candidates for 'AnakinSkywalker': ['https://swapi.co/resource/human/11']
Candidates for 'Tatooine': ['https://swapi.co/resource/planet/1']
Candidates for 'Amidala': ['https://swapi.co/resource/human/35']
Candidates for 'Watto': ['https://swapi.co/resource/toydarian/40']
Candidates for 'Fett': ['https://swapi.co/resource/human/22']
Candidates for 'Anakin': ['https://swapi.co/resource/human/11']
Candidates for 'Dooku': ['https://swapi.co/resource/human/67']
Candidates for 'MaceWindu': ['https://swapi.co/resource/human/51']
Candidates for 'Mace': ['https://swapi.co/resource/human/51']
Candidates for 'Windu': ['https://swapi.co/resource/human/51']
Candidates for 'PadméAmidala': ['https://swapi.co/resource/human/35']
Candidates for 'Coruscant': ['https://swapi.co/resource/planet/9']
Candidates for 'Skywalker': ['https://swapi.co/resource/human/11']
Candidates for 'Yoda': ['https://swapi.co/resource/yodasspecies/20']
Candidates for 'Naboo': ['https://swapi.co/resource/planet/8']
Candidates for 'BobaFett': ['https://swapi.co/resource/human/22']
Candidates for 'Padmé': ['https://swapi.co/resource/human/35']
Candidates for 'Boba': ['https://swapi.co/resource/human/22']
```

Figure 4: Star Wars entity retrieval and knowledge base setup for model training

tems we've tried such as MinIE, which performs poorly in sentences with negation.

3.5 Entity Linking

Entity linking for our purposes links an ambiguous entity to a known entity in order to augment information in a knowledge base. In the task of expanding an existing knowledge base, there is a need to determine in what context a particular entity is referring to. For example, given an entity with a partial name such as 'Skywalker', an entity linking model could determine that 'Skywalker' refers to Q1234, referring to 'Anakin Skywalker' or Q4321, referring to 'Luke Skywalker' depending on the context 'Skywalker' is used in a sentence. This aids in contexualizing new entities in order to determine the insertion point of the new triples into the existing knowledge base.

We begin the entity linking process by gathering the entities and its corresponding descriptions to train the linking model using the spaCy[10] custom training pipeline. We set a hyperparameter for the number of entities to train such as '10' and process the original dataset. We perform NER using spaCy and for every named entity that is identified and that also exists in the existing knowledge base which we want to augment, we add the entity to the collection. The descriptions are small bodies of text that describe the entity. We use these descriptions to generate a word embedding using BERT contextualized embeddings in order to train the linking model.

Next we set up the spaCy knowledge base using the vocabulary from the large English model and entity vector lengths of '300'. For each entity and description in the collection, we add the entity and the encoded BERT embedding of the description into the knowledge base. We create aliases in the knowledge base and their corresponding probabilities as well. The probabilities represent the proba-

[10]https://spacy.io/api/entitylinker

bility that a specific query will result in the entity in question. For example, given a query for 'Skywalker', the probability the query is referring to 'Anakin Skywalker' and 'Luke Skywalker' is both 0.50. To generate the training data for the linker model, we process the original dataset containing the sentences of all of the articles, and perform NER to identify if the particular sentence contains the entity we wish to train. Given this, we collect sentences over a threshold hyperparameter, for each entity we wish to train the model for. For example, given 10 entity and a threshold of 10, we collect 100 sentences total to train the model. During the training stage, we train the entity linking model with an 80% training, 20% testing split. We train for 500 iterations defining a minibatch size of 32, dropout regularization of 0.5 and a stochastic gradient descent optimizer. After training, we save the model for later use during the triple extraction and graph augmentation phase. One limitation to point out is that the entity linker model will have limited coverage of entities due to the non-completeness of the existing knowledge base.

3.6 Graph Augmentation

The graph augmentation step expands an existing knowledge graph with the new triples extracted from the unstructured text in the articles dataset. First we load the trained entity linker from the previous step and process the dataset to extract triples. For each triple extracted, we use the entity linking model to determine the QID of the entity and use that namespace for the final set of triples. For entities that are not trained by the model, we create a custom RDF URI from rdflib[11] such as 'www.example.com'. After processing the triples, we also perform a pruning step that eliminates unnecessary triples that do not provide quality information to the knowledge base. We simply check the subject and object entities of each triple and check if it contains stop words or any punctuation, and throw out the sample. This results in higher quality output. We finally augment the existing knowledge base by attaching a URIRef for each entity and relation in the triple.

4 Experiments

4.1 Dataset Creation

We generate an unstructured text dataset using the Dataset Creation module as described in the methods section. The user of this system simply inputs the start topic and the number of topics to collect. For this datasets, we use: (1) startTopic='Star Wars', numOfTopics=100. We collect 100 articles from wikipedia from 100 different topics resulting in 21253 sentences. Example output for the topic generation can be seen in Figure 3. After preprocessing, the StarWars dataset is 16521 sentences in total. We generate additional datasets for testing which can be accessed here[12].

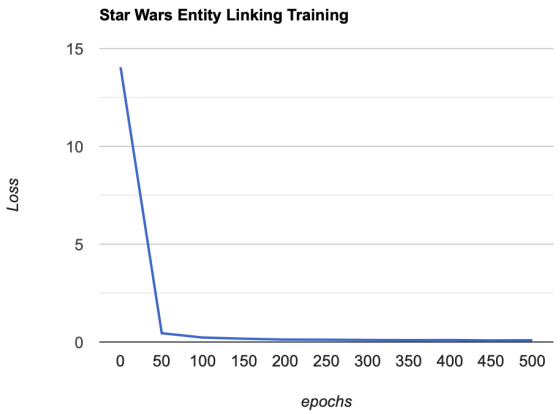

Figure 5: Entity Linking Loss Curve for Star Wars

4.2 Entity Linker Training

We train an entity linking model for the Star Wars dataset in the goal of augmenting an existing Star Wars knowledge graph 'starwars.ttl'. As seen in Figure 4, we retrieve the entities for training and set up the spaCy knowledge base for each entity and alias. We generate training data for the model training by collecting 10 samples for each of the entity we wish to train. The loss curve can be seen in Figure 5 where after 50 epochs, the loss converges. We evaluate the model performance on the test dataset to get around 85.71% accuracy on samples that the model has not been trained on. We also perform a qualitative analysis on an example sentence: 'Skywalker, also known as Darth Vader, is a fictional character in the Star Wars franchise.' The trained model predicts that the entity 'Skywalker' refers to the QID 'human/11' which refers to the character 'Anakin Skywalker'. In the context of the sentence, we can see that this prediction is correct because the character Anakin Skywalker is in fact referred to as Darth Vader. Hence, the entity 'Skywalker' has been disambiguated.

[11] https://github.com/RDFLib/rdflib

[12] https://git.elsys.gtri.org/projects/SERD-MADB/repos/ipykgb

Subject	Relation	Object
example.com/TheFilm	example.com/stars	example.com/EwanMcgregor
swapi.co/resource/human/35	example.com/develops	example.com/aSecretRelationship
example.com/TheGalacticRepublic	example.com/isThreatenedBy	example.com/aSeparatistMovement
swapi.co/resource/planet/1	example.com/accompany	swapi.co/resource/human/35
swapi.co/resource/human/11	example.com/fallTo	example.com/TheDarkSide
example.com/TheVillainRoster	example.com/includes	swapi.co/resource/human/11
example.com/TheLightsaber	example.com/belongingTo	swapi.co/resource/human/11
swapi.co/resource/yodasspecies/20	example.com/isVisitedBy	example.com/TheSpirit
example.com/TheCloneWars	example.com/releasedIn	example.com/march2011
example.com/StarWars	example.com/debutedOn	example.com/may251977

Table 1: Triple Examples for Star Wars Ontology

4.3 Triple Extraction and Graph Augmentation

For the final step, we extract triples from the dataset using relation extraction and the trained entity linker. For entities that are not covered by the linker model, we create a custom namespace and add it to the collection of triples. An example of triples from the Star Wars dataset can be seen in Table 1. During this step, we extract 54502 triples. After pruning, we extract 39030 triples. During graph augmentation, we expand the starwars.ttl knowledge graph from 4,029 triples to 41,399 triples

5 Conclusion and Discussion

In this work, we developed the framework for constructing knowledge graphs from unstructured text documents using various Natural Language Processing and Machine Learning systems. We aimed to solve both problems of constructing graphs from scratch and augmenting existing knowledge graphs with new triples in order to achieve more complete coverage of a particular domain. This entire process requires minimal human supervision and consists of four stages: Dataset Creation and Pre-processing, Term Extraction, Relation Extraction, and Entity Linking. Given a set of input unstructured text documents, dataset creation and preprocessing aims to automatically collect text from auxiliary sources such as wikipedia and standardizes the text corpus by removing noise. In the term extraction phase, we utilize Named Entity Recognition and Coreference Resolution systems to resolve entity dependencies in the collection of sentences. In the relation extraction phase, we utilize an OpenIE system to discover relations between entities within corpus to build out a set of triples to either construct a new knowledge graph or augment an existing one. In the entity linking phase, we train a neural network model to link an ambiguous entity to a known entity using IDs in order to augment more contextualized information to the knowledge base.

Future areas of work include improving the entity linking training model to cover a wider variety of entities to be trained on. In our current implementation, we automatically generate the training data by leveraging the original text corpus and perform NER to identify descriptions. This can be improved further by collecting more descriptive text for each entity to be trained on by using the dataset creation module to collect more articles about a entity. This will ultimately increase the accuracy and precision of the trained entity linking model and increase the coverage of entities that it is able to recognize. Another area of improvement is in the triple generation step. Currently we prune the output triples by filtering based on whether the subject or the object in the triples contain stop words or punctuation. A more sophisticated method of pruning will be needed to establish quality over quantity. Our current implementation expands an existing knowledge graph by 10 times the number of triples, which need to be reduced to avoid adding information that do not have semantic meaning.

In regards to tackling the non-completeness issue with training the entity linking model, we can achieve further completeness by adding a human in the loop to generate the training data for all of the entities in the existing knowledge base. By providing a UI that serves to match new entities with existing entities, we can generate a larger list of candidates to train the model. This will ultimately reduce the chances of training low quality entities and focus on those that provide the most

desired information in the domain. We can further add human supervision in the knowledge graph construction process by facilitating the triple pruning process. By allowing domain experts to select triples with the most description and quality, we can ensure that the output is optimal. This process will of course deviate from the fully automatic vision however, this will result in higher quality output, hence there is a trade-off to be made between quality and automation. Overall for this work, more experiments will need to be performed to assess the applicability of our work in downstream applications where knowledge graphs are needed to be generated automatically.

References

Muhammad Wasim Muhammad Usman Ghani Khan Waqar Mahmood Asim, Muhammad Nabeel and Hafiza Mahnoor Abbasi. 2018. A survey of ontology learning techniques and applications.

Hesham A Hassan Dahab, Mohamed Yehia and Ahmed Rafea. 2008. Textontoex: Automatic ontology construction from natural english text.

Shaoxiong Ji, Shirui Pan, E. Cambria, P. Marttinen, and Philip S. Yu. 2021. A survey on knowledge graphs: Representation, acquisition and applications. *IEEE transactions on neural networks and learning systems*, PP.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*.

Steven Bills Rion Snow Mintz, Mike and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data.

Yoav Goldberg Shwartz, Vered and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method.

K. Chen G. S. Corrado T. Mikolov, I. Sutskever and J. Dean. 2013. Distributed representations of words and phrases and their compositionality.

Jia Wu Xiaoyi Fu Jiachen Li Peng Zhou Wu, Xindong and Xu Jiang. 2019. Automatic knowledge graph construction: A report on the 2019 icdm/icbk contest.

Xindong Wu, Jia Wu, Xiaoyi Fu, Jiachen Li, Peng Zhou, and Xu Jiang. 2019. Automatic knowledge graph construction: A report on the 2019 icdm/icbk contest. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1540–1545.

Seunghak Yu, Tianxing He, and J. Glass. 2020. Constructing a knowledge graph from unstructured documents without external alignment. *ArXiv*, abs/2008.08995.

Seunghak Yu, Tianxing He, and James Glass. 2021. Autokg: Constructing virtual knowledge graphs from unstructured documents for question answering.

Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. 2021. Neural, symbolic and neural-symbolic reasoning on knowledge graphs.

Table 2: Opportunities/Challenges for proposed Methods and Systems

Task	System	Opportunities	Challenges
Dataset/Preprocessing	Wikipedia API	Automatically generates a list of articles given a topic	Suffers from disambiguation errors which need to be handled
	Topic Generation	Topic suggestions are consistent and semantically meaningful	Generates unrelated topics with deeper walks
	NP-chunking	Joins phrases and words together to avoid losing semantic meaning such as names	Often results in large chunks of text
Named Entity Recognition	spaCy lrg_English	Recognizes entities of varying types such as Date, Event, Work of art	Does not perform as well on larger bodies of text
	Stanford NER+NLTK	Performs well on Persons and Locations	Misses out on other complex entities
	Stanford NER+spaCy	Slightly better performance as compared to with NLTK tokenizers	Breaks entities into partial fragments
Coreference Resolution	Stanford CoreNLP	Performs well on larger bodies of text and utilizes dependency paths to match entities	Requires massive overhead to set up server and processing takes unreasonable amount of time
	Neural Coref	Requires little overhead to set up, model inference takes little time with reasonable accuracy	Word wise replacement of coreferences make little semantic sense in some scenarios
Relation Extraction	Stanford OpenIE	Extracts more relations in finer-grain detail	Takes some overhead to set up server
	MinIE	Little overhead to set up	Often misses obvious triples and performs poorly with sentences with negation
	Hearst Patterns	Little overhead, patterns provide some level of automation	Predefined patterns do not cover the breadth and complexity of language, performs poorly on large text
	OpenNRE	Neural method provides fast inference, can provide specific entities to extract relations for	Models are trained on predefined classes which do not cover the complexity of language
Entity Linking	spaCy training	Customizable, provides framework for training	Need to gather training data