A Task-Agnostic Machine Learning Framework for Dynamic Knowledge Graphs

Nicholas Sendyk*
Carleton University Faculty of Engineering and Design
Ottawa, Ontario, Canada
nicholas.sendyk@disroot.org

Curtis Davies*
Carleton University Faculty of Engineering and Design
Ottawa, Ontario, Canada
curtis.davies@riseup.net

Titus Priscu†
Carleton University Faculty of Engineering and Design
Ottawa, Ontario, Canada
tituspriscu@cmail.carleton.ca

Miles Sutherland†
Carleton University Faculty of Engineering and Design
Ottawa, Ontario, Canada
milessutherland@cmail.carleton.ca

Atallah Madi†
Carleton University Faculty of Engineering and Design
Ottawa, Ontario, Canada
atallahmadi@cmail.carleton.ca

Kevin Dick
Carleton University
Ottawa, Ontario, Canada
kevin.dick@carleton.ca

Hoda Khalil
Carleton University
Ottawa, Ontario, Canada
hoda.khalil@carleton.ca

Ala Abu Alkheir
Lytica Inc.
Kanata, Ontario, Canada
ala_abualkheir@lytica.com

Gabriel Wainer
Carleton University
Ottawa, Ontario, Canada
gwainer@sce.carleton.ca

ABSTRACT

Many applications require well-structured and current information to enable downstream tasks. Knowledge graphs are a type of knowledge representation that effectively organize current information capturing elements and the relationships between them such that they can be queried and/or reasoned over in more advanced applications. A particular challenge is ensuring that an application-specific knowledge graph is both comprehensive and contains the most current representation, achieved through dynamic updating. Some available software frameworks for managing information as part of a data science pipeline are effective in collecting, labelling, and analysing textual data using natural language processing. Despite the utility of these frameworks, they can nonetheless be daunting for use by industry professionals and/or researchers who may not be familiar with the specifics of each tool. In this work, we present a generalized task-agnostic supervised machine learning framework that serves as a streamlined methodology for the creation and dynamic updating of knowledge graphs. A user needs only to define task-specific parameters allowing the tool to scrape data from the internet, generating a candidate corpus. The user may then provide sample annotations from the corpus to train task-specific natural language processing models to extract the relevant knowledge graph elements and the relationships connecting them. We demonstrate the utility of this framework for a case study seeking to build knowledge graph representations of merger and acquisition events between companies from scraped online articles reporting these instances. Our task-specific machine learning models achieve upwards of 99.2% F1 score evaluation metric on candidate web page classification and 81.5% F1 score on sentence-level extraction of entity relationships, demonstrating the promise of this framework. Our framework is freely available at: github.com/Checktr/tadkg.

*Both authors contributed equally to this research.
†These authors contributed equally to this research.

CCS CONCEPTS

• **Applied computing** → **Document management and text processing**; **Enterprise data management**; • **Computing methodologies** → *Supervised learning by classification.*

KEYWORDS

data science, natural language processing, web scraping, machine learning, knowledge graph updating

ACM Reference Format:
Nicholas Sendyk, Curtis Davies, Titus Priscu, Miles Sutherland, Atallah Madi, Kevin Dick, Hoda Khalil, Ala Abu Alkheir, and Gabriel Wainer. 2023. A Task-Agnostic Machine Learning Framework for Dynamic Knowledge Graphs. In *Proceedings of CASCON 2022 (CASCON'22).* ACM, New York, NY, USA, 10 pages.

1 INTRODUCTION

In this fast-paced digital era where online data is continually evolving, methods to capture, represent, and analyse those data are needed to accurately model their underlying phenomena (and convoluted patterns that could indicate otherwise unknown correlations between objects). A number of large-scale (sometimes internet-wide) initiatives organize extracted information within a knowledge graph (KG) structure. KGs, in their most abstracted form, are comprised of nodes representing specific elements/entities and are related to one another by edges/connections representing information of that relationship [4]. Once constructed, KGs will contain large amounts of prior knowledge and are an effective means of organizing data enabling multitudinous downstream applications such as the creation of recommendation systems, search engines, and question-answering systems [3].

Examples of large-scale KGs include Google's Knowledge Graph [19], WordNet [14], YAGO [20], and Freebase [2]. In 2012, Google introduced its Knowledge Graph project, which sought to improve its search engine [19]. WordNet is a lexical database for the English language where graph nodes represent synsets (sets of cognitive synonyms comprised of nouns, verbs, adjectives, and adverbs) that are interlinked by means of conceptual-semantic and lexical relations enabling downstream applications such as machine translation, text classification, and text summarization [14]. YAGO is an open-source KG integrating information extracted from Wikipedia, WordNet, and GeoNames and attached both a temporal and spatial dimension to many of its facts and entities [20]. To date, YAGO contains knowledge of over 17 million entities with over 150 million facts about those entities [20]. Finally, FreeBase is a large-scale collaborative KG comprised of community-submitted information as well as data harvested from sources including Wikipedia, the Notable Names Database, the Fashion Model Directory, and MusicBrainz [2]. Individually and collectively, these KGs represent massive stores of information on specific elements, as well as on the relationships that connect them.

While generalized and massive-scale KGs are necessary for a breadth of downstream applications, the scope of a given KG can be considerably more focused for specific downstream tasks. Gathering online data for a particular set of elements and certain types of relationships that relate them is often necessary for more targeted tasks. Examples include matching a set of faculty members names to their home university, determining whether certain venture capitalists (VC) have invested in various companies, or determining whether a given company has undergone a merger or acquisition with another. In each instance, online information must be gathered from numerous differently-structured web pages and using adapted natural language processing (NLP) techniques, the relationships between those identified entities must be accurately extracted for integration into a dedicated KG. Essentially, extracting these specific relationships - through the use of Google Search, for example - may be roughly analogous to taking a particular *slice* from Google's own Knowledge Graph, however, those desired entities and/or relationships may not always be contained within a more generalized KG requiring specific frameworks to construct one's own. Moreover, as the underlying relationships evolve over time, the KG must also evolve to reflect those changes.

Whether generalized and massive-scale or task-specific and of modest-scale, critical to the utility of KGs is the *recency* (the "freshness") of their contained information. Consequently, once a KG is created, it also must usually be maintained through dynamic updating. For example, the Never-Ending Language Learning system (NELL) developed at Carnegie Mellon University is a semantic machine learning (ML) system that runs continuously (*i.e.* 24/7 and forever) in its task of learning to read and generate new facts from the web [15]. While certain information may be updated with relatively high frequency necessitating continual KG modelling, other processes (*e.g.* achieving university tenure, VC investment, company mergers & acquisitions) evolve at relatively slower frequencies implying that KG updates (including careful deduplication) need only occur at spaced intervals (*e.g.* once a month) to ensure recency in the resultant KG.

The task-specific generation and dynamic updating of KGs requires the coordination of a large diversity of software tools as part of an integrated framework. Also incorporated within these framework state-of-the-art NLP models that, through fine-tuning and/or transfer learning, are tailored to particular tasks. For example, the PubMed Knowledge Graph was created using a framework specifically tailored for leveraging Bidirectional Encoder Representations from Transformers for Biomedical Text Mining (BioBERT) [10] in order to extract bio-related entities from over 29 million PubMed abstracts [21]. Similarly, Amazon has sought to build a Product Knowledge Graph by harvesting product knowledge from semi-structured sources on the web and from text product profiles through the use of a complex series of deep learning models and methodologies [6].

Research groups, companies, and various organizations that might benefit from the generation of their own task-specific KG are unlikely to have the resources and expertise to develop the necessary software framework [11]. Given that many of the common procedures in generating and dynamically updating a KG can be abstracted into a few simple steps (summarized in Fig. 1), it would be of immense benefit to research groups, companies, and various organizations alike to make use of a task-*agnostic* framework that, through simple user parameterization, could be tailored to a specific task.

To that end, we introduce within this work, the first open-source, task-agnostic machine learning framework for dynamic KG generation and updating. Such a general framework can be easily adapted and reused to solve a variety of problems. The framework enables users to quickly develop systems that accurately extract online information, enabling the generation and dynamic updating of task-specific KG from the evolving web. We collaborate with an industry partner, Lytica Inc., to demonstrate the utility of the proposed framework. We apply the framework to a case study by creating and dynamically updating a KG of electronic component manufacturer merger and acquisition events.

2 FRAMEWORK DEVELOPMENT, DATA COLLECTION, & METHODOLOGY

When developing a dynamically-updated KG for a particular task, one of the greatest challenges is identifying the relevant online information with which to generate and update the KG. Projects focused on a niche topic are typically updated from online sources in a variety of unstructured forms. These data are not amenable to simple extraction, given that they may be embedded as text-based content interspersed among a variety of digital (meta)data. Examples of task-specific target sources may include news articles, press releases, blogs, and other forms of digital media. Gathering such information is a difficult task. Designing a solution that is specific for each task may be efficient, in some cases, but is time-consuming and redundant. An elegant task-agnostic solution can address these challenges while guaranteeing efficiency, re-usability, and higher quality through more testing opportunities.

The framework we present here facilitates rapid and accurate generation of dynamic KGs and functions such that it may update

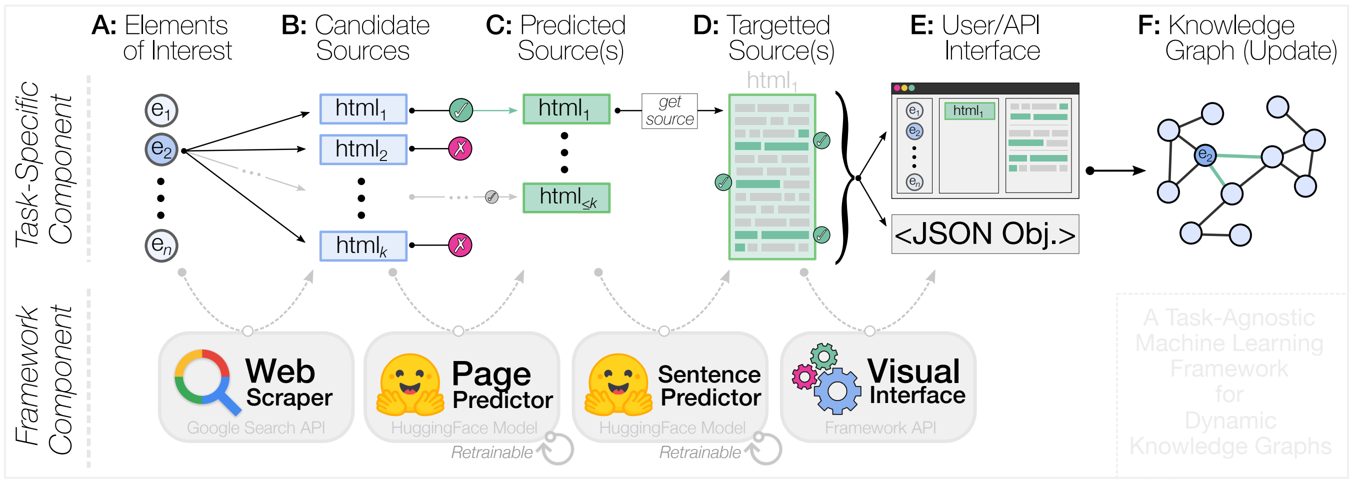


Figure 1: Conceptual Framework Overview applying the Machine Learning Framework to an Abstracted Task.

a given KG in near-real-time. The framework facilitates all aspects of task-specific data collection, data analysis, data enrichment, and NLP model generation, which together allow for the generation and deployment of highly scalable KG enabling various downstream applications.

2.1 Framework Overview

The framework we present abstracts the common steps in KG generation and updating into the four general components illustrated in Fig. 1. A given user can parameterize/modify each framework component to suit their specific application; the functions of each component are described below.

2.1.1 Web Scraping. The first component in our framework is a user-configurable web scraper that leverages the Google Search API. A generalized web search and scraping framework enables the handling of the diversity and unstructured formatting of potential search results. Users define an input set of candidate entities along with particular search terms to use in conjunction with those entities to narrow the web scraper to candidate web pages. While the Google Search API is an accessible way to gather web pages from a set of keywords, it may present a challenge with its financial cost. The framework attempts to mitigate this issue by separating the web-search and web-scraping components of the process to allow users to pass web page URLs through other methods, such as another search engine or a predetermined list of URLs from a file.

When a given web page is added to the scraping queue, it is first examined to ensure it is formatted as a HyperText Markup Language (HTML) file. Web searches can yield non-HTML results such as PDF files which search engines deem relevant to the searched topic. Non-HTML pages are determined by checking the `Content-Type` header in the web server's response and subsequently ignored within our framework. While non-HTML files may contain relevant information, many of them contain data that is encoded in a binary format, making text extraction a much more difficult process (*e.g.* require the application of Optical Character Recognition software).

Once the page format is verified, the HTML-encoded data is passed into the text extraction process to be scrubbed for information. The framework uses the Beautiful Soup [18] Python library to pull text from the encoded data to store it. If the framework configured to store the data as separate sentences, a regular expression is applied to detect sentence and paragraph splits within the text. Each piece of data, whether a full page (denoted *page-level* data) or a sentence (denoted *sentence-level* data), is stored in the appropriate training database.

Structured data within the framework is stored within SQLite [9] databases that are manipulated by different segments of the code. Using SQLite allows for easy sharing and backing up of data within the databases for data replication and collaboration during development and use.

The training database contains columns containing information that is relevant to the training of the downstream NLP models and for subsequent data analysis. These columns include headers for the URL of the web page from where the extracted text originates, the entity being examined and searched for to retrieve the page, the text itself and any user-defined parameters on which the NLP model is to be trained upon. Our framework support the manual annotation of these data to produce task-specific NLP models as part of the training and prediction phases, described next.

2.1.2 NLP Models - Page Predictor & Sentence Predictor. The framework uses multiple NLP models to handle unstructured textual data and to extract and/or classify particular textual elements. In order to generate task-specific classification models, our framework in its generic configuration uses a combination of Google's BERT [5] transformer model and HuggingFace's Longformer [1] model to classify text segments of varying lengths.

Users are provided with terminal-based annotation tools capable of providing users with textual examples to be classified according based on their relevancy to the particular task. By preparing training examples (usually about a few hundred), the NLP models can be adapted to identifying candidate web pages and textual data relevant to the user's intended task.

Applying transfer learning to these pre-trained models allows for context-aware analysis of arbitrary articles and sentences from the internet without needing to prepare the models from scratch for any specific topic. The framework, for data augmentation purposes, additionally leverages Huggingface's PEGASUS [22] model to generate synthetic data similar to sentences within the extracted database. When applied to individual sentences, the PEGASUS model can create new sentences of similar lengths and similar meanings using synonym words and different sentence structures sharing the same semantic meaning. Conveniently, PEGASUS is also leveraged to summarize the scraped articles into shorter documents while maintaining their semantic meaning.

In summary, the resultant page-level classification NLP model is capable of predicting whether a given candidate page is relevant (see Fig. 1B & C) and, subsequently, the sentence-level segmentation model NLP model identifies the candidate sentences within a given HTML page as relevant to the extraction task.

2.1.3 JSON Output & Visualization Interface. Finally, to provide an interactive interface to make transparent to the user all stages of the data extraction pipeline, a visual interface (and machine-friendly API) is available to the user to oversee the complete KG generation and updating process. The interface functions in near-real-time, enabling a user to enter new entities and trigger the pipeline to extract relevant, online, and task-specific information. The underlying API generates JSON objects that can easily interface with graph-generating software such as Cytoscape or Gephi (not included within this framework).

2.2 Task-Specific NLP Model Training Pipeline

In order to build the initial task-specific model(s) used to make predictions, the framework needs to apply transfer learning to a pre-trained NLP model using an initial dataset. The framework uses a supervised learning approach, requiring a set of sample inputs and expected outputs for the model to process and determine how to analyse future data to make predictions. This section of the work shall be referred to as the *Training Pipeline*. A detailed overview of this pipeline is illustrated in Figure 2.

2.2.1 Corpus Acquisition. To obtain an initial set of textual data to train the ML model(s), the framework performs web searches created from user-specified templates and keywords. Candidate search results are logged and their source-code is scraped to extract textual information before being added to the training database.

Within the framework's configuration file, the user specifies a list of entities to be searched for along with templates to insert these entities into. An example of this would be the entities *USA*, *Germany* and *Sweden*, with a template of "`Tourism in {} 2022`". Given these parameters, the framework replaces every token `{}` with an entity from the list. The usage of query templates over raw query strings allows for a much broader set of articles to be pulled with minimal configuration by the user.

With the list of search queries generated, the framework uses Google [16] to pull relevant web pages and articles. The user is able to adjust parameters passed to the Google Search library that may yield more optimal results, such as the region to search for or the number of results to pull per query. Through several rounds of experimentation, we determined that the default number of unique web pages returned per query would be set to $k = 25$. This decision balances both the coverage of candidate pages of putative relevancy with the risk of acquiring an excessive number of potentially irrelevant pages, which exacerbates class imbalance (true relevant web pages are generally assumed to be rare) and represents noise in our subsequently trained model.

In reference to Figure 2, this can be seen as the top portion of the model taking input arguments and then using linkgrabber.py to collect a list of relevant web pages. These web pages are then converted to text using thedrycleaner.py. Now that the text is obtained, they can be cleaned of unwanted text and put into the page text table so that only the raw data for each search is used for the rest of the training data.

2.2.2 Data Labelling. In formulating the web page prediction task using a supervised learning, users must provide labels to a sufficiently large subset of the initially extracted candidate textual data to fine tune a pre-trained NLP model in order for it to then make predictions based on future inputs. The framework provides an interactive tool that allows the user to label rows of data within tables, representing either entire articles (page-level annotation) or sentences (sentence-level annotation), based on parameters specified within the configuration file. Due to the nature of this labelling process, the concept of truth is dependent on the results of the labelling process by the user and takes no other externally sourced input into consideration.

A common problem with labelling within data science is the potential annotator bias introduced within the labelling process. For example, if articles and sentences are added into the database sequentially, a user labelling data in the same order of incorporation, bias may be unintentionally introduced into the training data set due to unbalanced proportions of labelled rows between entities. In order to minimize this effect, the labelling tool selects random rows from the database for the user to label, preventing unconscious bias stemming from the order of records within the data set. On average, the proportion of labelled data will approximate the proportion of representation a given entity has within the complete data set. For many applications, such an approach should be sufficient, however users applying this framework to applications exhibiting extreme class imbalance (long-tailed distributions) are encouraged to adapt the labelling strategies through appropriate means (*e.g.* stratified sampling, over-/under-sampling).

Looking at Figure 2 it can be seen that in the middle and bottom lines there are 2 labeling sections. The middle row labels at a page level to tell the model whether the pages taken from the page text table database contains information of use. The pages that are found to be useful, labeled with "1", are then copied over and split into sentences into the sentence text table database. Now the bottom row of the model labels from the new database created to find which sentences specifically contain useful information to the user.

2.2.3 Synthetic Data. In some cases, data pulled from web searches does not contain enough positive data to create a model that will yield accurate results pertaining to the user's intended topic. Without sufficient data to be flagged as positive, the model will incorrectly label many results as negative and yield poor accuracy. Our

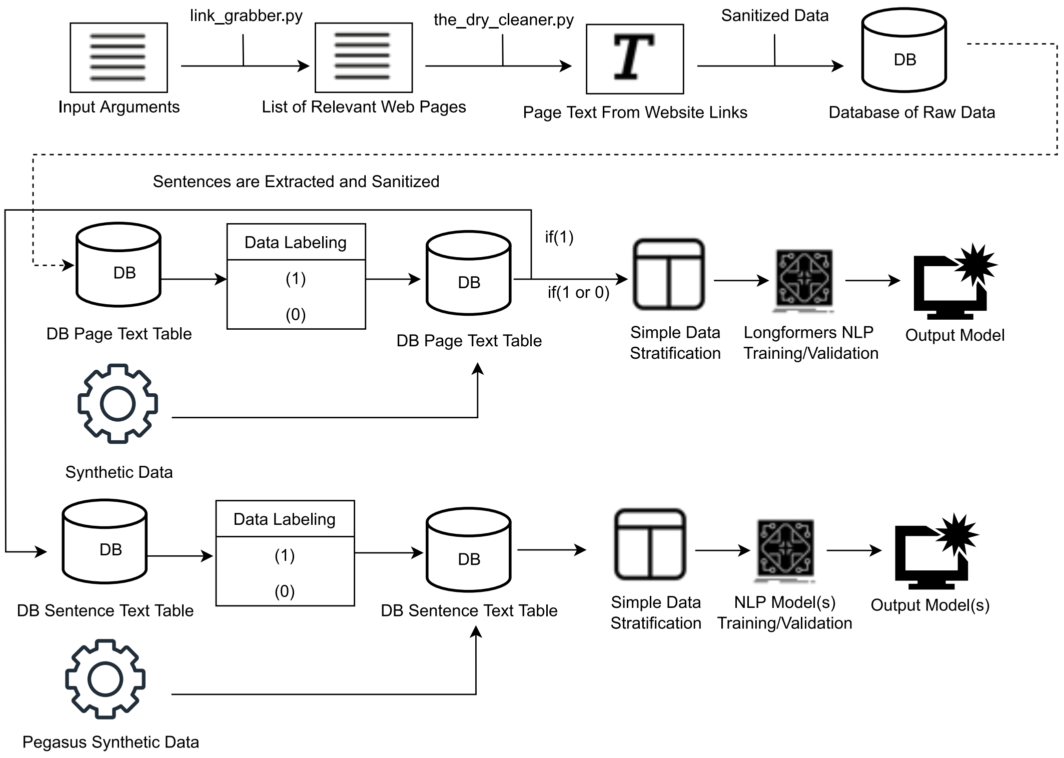


Figure 2: Detailed Overview of the Framework's Training Pipeline. Data is acquired using input arguments given by the user. The data is then manually labeled and models are trained to use in the prediction pipeline.

framework provides a complimentary function to generate synthetic textual data with specific output values specified by the user. This allows the model to more accurately identify patterns to be labelled as positive, even when genuine data pulled from web searches provides minimal information on these sentences.

In a similar format to creating search query templates, the user can organize "fake entities" and templates making sentences similar to what the framework would find when scanning articles representing specific kinds of information. The results generated from each template are spun using the PEGASUS [22] summarization model. This allows for further iterations of each synthetic data point, leading to a broader, less biased set of sentences upon which to train our task-specific model(s). This synthetic data is used alongside genuine labelled data to build a more accurate model to detect the requisite KG relationships.

During the initial development of our case study, our experiments demonstrated that the inclusion of synthetic data to the existing training data set increased model F1 scores by up to 12%, considerably improving model accuracy. In order to preserve transparency, the synthetic data is marked as such within the database and may easily be filtered out in subsequent applications.

Once the data is labeled and put back into both the page text and sentence text table databases, synthetic data is also added to the sentence-level table. This process can be seen in Figure 2, where labelled data and synthetic positive data are combined pre-stratification and used to train the model on the third row of the diagram.

2.2.4 Model Training. Once the data set has been curated and annotated (with and/or without the inclusion of synthetic data), the NLP model(s) are trained. The first stage of the training pipeline process is data stratification. Depending upon the bias of the collected data, or the number of classes within it, training an NLP model can become confounded by those biases, resulting in poor results. One solution to circumvent this is to refine the training data to a high-quality stratified subset of samples with which to train the model. This approach risks removing a portion of the data depending on the stratification method chosen, but the end result generally produces a much more refined model.

Using the stratified data, the framework leverages transfer learning of HuggingFace models through supervised learning to train models. The specific models chosen to be trained are context-dependent on the use case, as many problems have a variety of needs and classifications associated with those needs. The specific models leveraged in our task-specific case study are discussed below. The trained models are then used to make predictions in the Prediction Pipeline.

The Final step in the model in Figure 2 is the training of the models. The page level training in the middle of the figure takes the database containing the labeled data and takes it to be stratified.

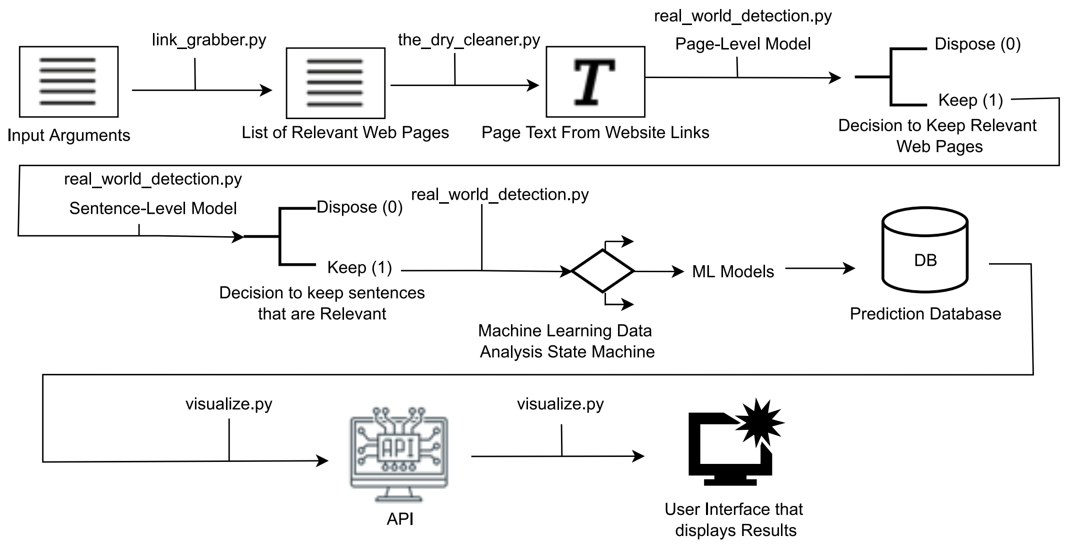


Figure 3: Detailed Overview of the Framework's Prediction Pipeline. pipeline gathers data with reference to user input arguments make prediction and output results. The data acquired is pushed through the multiple models made with the already manually labeled data. The pipeline than creates a visualization of the data for the user to use and analyze the results.

the data then goes through the Longformers NLP training and the model is output. for the sentence level the process in very similar. The labeled sentance level data is stratified and trained to create the output model.

2.3 Task-Specific NLP Prediction Pipeline

Once the Training Pipeline has completed, the necessary resources are organized for the general deployment as part of the *Prediction Pipeline*. The framework uses the NLP classifiers trained in the initial pipeline and formats these models according to the user-configured requirements for specific KG generation or update. This configured script is used to make predictions on newly extracted data and ultimately be integrated into the resultant knowledge graph for downstream applications and/or data analysis. A detailed view of the make-up of this pipeline is shown in Figure 3.

2.3.1 Data Gathering. Gathering raw information for prediction is done in a manner akin to the Corpus Acquisition found in section 2.2.1. Textual data is gathered via the framework through web searches according to user-specified templates and keywords. The resultant data are then logged, web pages scraped, and the textual data is extracted and stored. The framework can be configured to specify the queryable entities through multiple schemes; users can optionally provide a supplied list of entities or interact with the interface dynamically by providing a single entity at a time. There is no inherent cap on the length of the list of entities.

The results gathered from each search are collected into a Panda's Data Frame. These results are then sanitized and formatted so to be consistent with the training data. This Data Frame containing sanitized results is then passed through the pipeline for prediction by each of the fine-tuned models.

In figure 3, the acquisitions of links is similar to that in Figure 2. The pipeline takes input arguments and using linkgrabber.py and thedrycleaner.py acquires the text from the relevant web pages.

2.3.2 Multi-Model Predictions. Making predictions on even a modest subset of online sources is a computationally demanding task. Employing a "divide-and-conquer"-like approach is necessary to avoid wasteful application of compute resources on irrelevant data.

The default framework considers two general levels of data as part of the Prediction Pipeline, the page-level representation for candidate web pages and the sentence-level representation for candidate entities and relationships. Our framework is configurable to allow for additional levels and types of abstractions, in its generalized/default form, the two page-/sentence-levels of abstraction should be sufficient for the majority of tasks.

The page-level of abstraction represents a sanitized candidate web page to the page-level NLP for classification. If rejected as irrelevant, we spare considerable computational expense to further consider the sentences contained within. Using this approach, our framework has demonstrated great success implementing the Longformers [1] pre-trained model for our case study application (described below).

If a candidate web page is instead classified as relevant, the sentence-level model is applied to the contents of that page, split by the framework into sentences and each classified for relevancy. Should the sentence be classified as containing relevant information to the KG, it is classified as a positive. The sentence-level models can be implemented as a binary classifier, named entity recognition model, or any number and mix of other NLP model which is relevant to the user-defined project.

Once the data is acquired, the data goes through the page level and sentence level models created in the training pipeline. Following Figure 3, with the page text from the website links it is then put through the realworlddetection.py page level model. through the model if the link shows relevant data the page is marked with a "1" and if not it is marked with a "0". Links with "1", continue to the sentence level model and links with a "0" are disposed as they do not provide relevant information. The process then happens again with sentence level model for the data that passed through the page level model.

2.3.3 Output. The Prediction Pipeline results are stored in an SQLite database, separate from the training data. This database contains results from the pipelines along with metadata on the process itself, including timestamps and confidence scores.

This data is made readily accessible through an interactive web-based dashboard, listing entities and the web pages associated with the results detected by the framework (Fig. 1E). An example of this dashboard for the predictions specific to our case study is shown in Fig. 8. Results are also made accessible through an HTTP REST API, formatted as JSON semi-structured data. The API allows for specific queries to be performed on the data set, such as searching for specific entities and searching the web for results based on new entities in near-real-time (about seconds to minutes). Due to the nature of the model's ability to recognize entities based on patterns found within sentence structures, a model trained on well-labelled data will be able to efficiently learn new facts about previously unknown entities at runtime. This API-based method of querying data leads to minimal load on a client fetching data from the data set, allowing for lightweight, remote access to results from the models on any custom client organized by a user. Ultimately, the visualized results and/or JSON data can be easily integrated into an existing KG and visualized using conventional network visualization software.

After going through both models the data is then pushed through the Machine Learning Data State Machine where the results for each page are processed and then stored in the Prediction database. Now at the bottom of Figure 3 it is seen that the data lastly goes through visualize.py and the API to output the User interface containing the results of the prediction pipeline.

3 CASE STUDY - MERGER & ACQUISITION EVENTS

A case study for the utility of this framework was performed in collaboration with Lytica Inc. [12]. Lytica Inc. provides risk management services to companies looking to purchase electronic components, collecting data on the electronic component manufacturer supply chains and providing relevant information allowing companies to make informed purchasing decisions.

To maintain a KG of existing electronic component manufacturers (and the relationships between them), Lytica Inc. sought to deploy a system to intermittently detect mergers and acquisitions between these electronics manufacturers companies. While databases of similar information exist as paid services from other companies [13], Lytica Inc. sought to integrate their own configurable KG generation and updating service that may potentially detect mergers and acquisitions not captured by competing services. The requested solution would search the internet for merger and/or acquisition events pertaining to a list of predetermined companies. Lytica Inc. provided a list of 200 example company names to test the system against and which represented the initial set of entities to query.

The system was configured to generate results indicating whether a company was involved in a `merger` or `acquisition` or `merger acquisition`. The web scraping component was set up to generate queries searching for mergers and acquisitions related to a portion of the companies listed. Synthetic data with falsified or unrelated company names were also generated to avoid bias towards specific subsets of the electronic manufacturers provided. Data was collected and generated at both the page- and sentence-level to allow for extensive training of each model.

With the data prepared, the system was configured to train four separate classifiers with PyTorch [17] and Longformer [1] models pertaining to the presence of mergers and acquisitions with each company. A single classifier, *PageLevelClassifier*, determined whether a web page contained information relevant to a merger or acquisition. If this model deemed a page relevant, it was split into sentences and passed to the *MergerOrAcquisitionClassifier*. This classifier served as a general purpose sentence-level classifier to determine if a particular sentence had any relevant information for a merger or acquisition. A sentence passing this classifier was passed to the *MergerClassifier* and the *AcquisitionClassifier* to determine the likelihood of whether the relevant sentence contained a merger, acquisition or both.

Each model's training session provided a confusion matrix containing metrics derived from the rate of True Positive (TP), True Negative (TN), False Positive (FP, Type 1 Error) and False Negative (FN, Type 2 Error) hits when testing the model against specified testing rows within the training database.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{2}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$FalseNegativeRate = 1 - Sensitivity \tag{5}$$

$$FalsePositiveRate = 1 - Specificity \tag{6}$$

A Type I Error occurs when the model predicts that an acquisition/merge did happen, when it did not. Conversely, a Type II Error occurs when the model predicts that an acquisition/merge did not happen, when in actuality it did. These metrics are used to compute various indicators that can be individually and/or collectively considered to evaluate model performance and reliability. The results extracted from the confusion matrix of each of the classifiers are tabulated in table 1.

4 RESULTS AND DISCUSSION

In order to lower the barrier of entry to KG generation and updating for individuals, the research community, companies, and various organizations, an open-source and user-customizable framework is required. While the space of potential highly-specialized use-cases

Table 1: Results of Classifiers

Classifier	Merger	Acquisition	Merger or Acquisition	Page level
Accuracy	0.98	0.98	0.98	0.81
Precision	0.99	0.98	0.99	0.86
Sensitivity	0.985	0.98	0.9960	0.7701
Specificity	0.9818	0.98	0.9408	0.86
False Negative Rate	0.0195	0.0095	0.0040	0.2299
False Positive Rate	0.0182	0.0396	0.0592	0.14
F1 score	0.9877	0.9896	0.9920	0.8146

for such a framework cannot conceivably be covered by a single, task-agnostic framework, we consider that numerous applications could be simplified to resemble the formulation of our use case. Thus, integrated within our framework defaults are many of the experimental values deemed best suited to a general KG generation application.

4.1 Case Study Results

As part of the mergers and acquisition event detection case study for the framework presented within this work, we note that highly accurate NLP models were created and the resultant KG reflects well the underlying merger and acquisition events in the electronic component manufacturer space. In this work, over 400+ web pages and 1,500+ sentences were manually labelled. This provides a general sense for the magnitude of annotation effort required to achieve a comparable level of performance. As is common with machine learning systems, it is our expectation that the greater the number of consistent and high-quality annotations produced, the greater the subsequent models will perform in their targeted task.

The `Merger` classifier results can be seen in table 1 and can be seen to have produced very accurate results with an F1 score of 0.9877. The classifier had very low false negative and false positive rates, indicating that the merger classifier is very efficient at finding mergers within the text of the sentences.

Relative to the Merger classifier, the false negative rate is lower for the `Acquisition` classifier, mainly in part because the data that was labelled had more instances where companies would outright acquire other companies rather than merge with them. As a result of this domain-specific knowledge, it is expected that the false negative rate be lower in the Acquisitions' classifier compared to the Merger classifier, since the semantically similar terms for "merger" would not be captured in the resultant candidate web pages and sentences.

The web page classifier did not produce as strong results relative to the Merger and Acquisition classifiers due to two main reasons. The first reason is that fewer web pages were labelled at the same time compared to the sentence level. There were approximately 400+ web pages manually labelled, compared to the 1,500+ manually labelled sentences. The second reason is that the web page's text sometimes contains over 10,000 words. The classifier originally used was BERT [5] which could only analyse 512 words. Then, we implemented the Longformer [1] classifier, which could analyse 4,096 words at optimum computations. However, on average the limitations set by the computers used allowed for 3,072 words only to be analysed. Even with the increase in word input from BERT [5] to Longformer [1] methods, this caused the page level classifier to have higher false positive and negative rates. This challenge could be mitigated by explicitly splitting candidate web pages into "batches" sufficiently sized for page-level processing, in order to explicitly conserve all textual information.

The last sentence-level classifier that was trained was the `Merger or Acquisitions` classifier. Naturally, since this classifier looks for both cases of merged or acquired, it would be expected that the false negative rate would be lower. Which was observed, the false negative rate was 0.0040 which is less than 0.0195 (Merger) and 0.0095 (Acquisitions). Conversely, the false positive rate would be higher since the classifier is detecting both cases at 0.0592 which is higher than 0.0182 (Merger) and 0.0395 (Acquisitions). It is important that this classifier has the lowest false negative rate and highest false positive rate to act as a second filter process for the text from each web page. By applying the `MergerOrAcquisition` classifier before both Merger or Acquisition classifiers results in the most accurate and precise decisions for the final output that the user would see. Additionally, the `MergerOrAcquisition` F1 score was the greatest at 0.9920 while the web page-level F1 score was 0.8146.

4.2 Visualizing the Knowledge Graph Results

Following the use of the framework, a rich, dynamically-built dataset was produced representing a KG of merger and acquisition events between electronic component manufacturers. This data set enables downstream applications through querying. To more explicitly visualize some of the resulting relationships, network plotting libraries were used.

Figure 4 visualizes a graph to show, for a subset of the data collected, every relationship between entities in the resulting data set. Through the use of network-based metrics, additional relationships and insights may be derived (*e.g.* are there nodes with abnormally high in-degree representing a conglomerate of other manufacturers?). As shown in Fig. 5 multiple criteria can be used to filter and gather all known relationships relating to mergers and acquisitions. This visualization (albeit not interactive in this form) is highly valuable for exploring the information and the connections between entities and their surrounding mergers and acquisitions.

Figures 6 and 7 show more actionable insights. For the given subset of data, these graphs demonstrate specific actionable search criteria. Using such language filters (*i.e.* in this situation the filters of "acquired by" and "acquires") specific relationships can be extracted from the resulting data set produced from our framework. These types of insights can be used to update existing KG, or to be used as the basis for further data exploration.

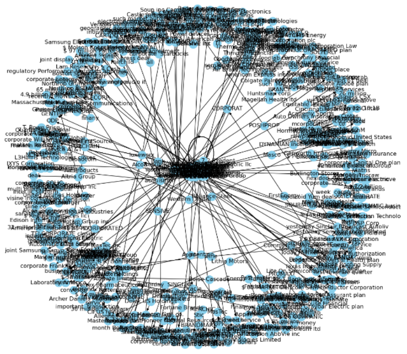


Figure 4: Graph Visualization of All Extracted Relationships

Figure 5: Graph Visualization of "Merger or Acquisition" Relationships

Figure 6: Graph Visualization of "Acquired By" Relationship

4.3 Training Data Enrichment is a Major Challenge to Knowledge Graph Generation

Knowledge Graph generation applications require multitudinous unstructured raw data and enriching these data (*i.e.* processing, sanitizing, and labelling) poses a significant challenge; most raw data is usable in its unprocessed format. In order to use sampled data for the training of NLP models, data must first be formatted and labelled accurately. There does not exist a one-size-fits-all approach to preparing these data for all user-specific tasks. Our recommendation is to adhere as closely to best practices [7] and to consider having multiple independent annotators label the same data to produce a measure of inter-rater reliability estimated using Fleiss' weighted kappa [8].

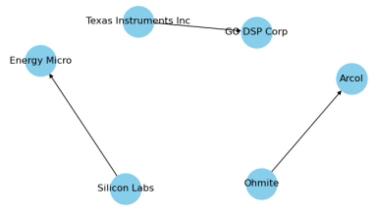


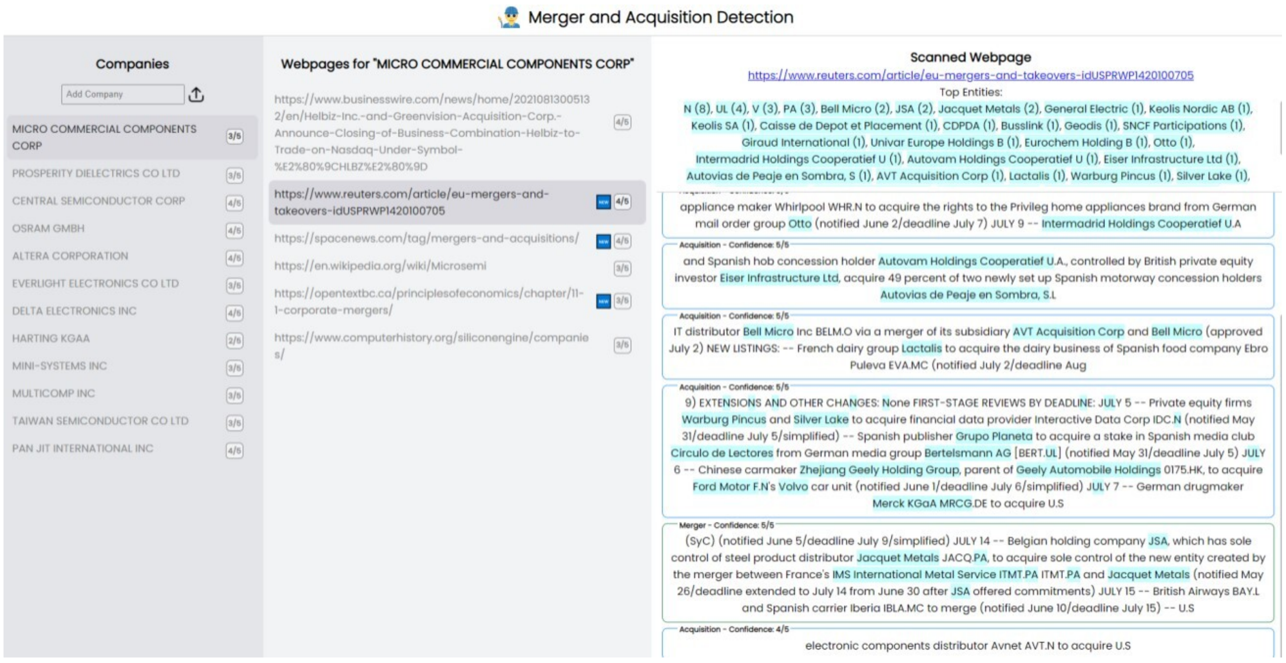
Figure 7: Graph Visualization of "Acquires" Relationship

4.4 Further Expansion of Framework

While the presented framework promises broad applicability, there yet exist numerous avenues in which the work can be expanded. While the present case study represents a small-to-moderate sized application of our framework, the scaled application to considerably larger inputs has not been demonstrated nor evaluated. The computational complexity of our framework is challenging to define, given its intrinsic task-agnostic nature. Further computational benchmarking of the framework would benefit end-users seeking to apply the framework to their particular problem. The subsequent identification of putative computational bottleneck may further improve our framework. Nonetheless, we anticipate that our framework in its present version holds considerable promise for a wide array of applications in disparate fields such as sociology, criminology, and finance where end users may benefit from a lowered barrier to entry to the creation and querying of their own task-specified knowledge graph.

5 CONCLUSION

In conclusion, the presented framework provides, for the first time, an open-source and standardized method for individuals, research communities, companies, and various organizations to build and maintain dynamic knowledge graphs. The framework provides a means for individuals to quickly, at low cost, and with low effort generate a knowledge graph based on task-specific configurations. As exemplified in a case study in collaboration with Lytica Inc., our framework succeeded in producing highly accurate language models and subsequently produce a large-scale KG representative of merger and acquisition events among electronic component manufacturers. Frameworks such as the one developed here hold significant potential in reducing the barrier to entry for non experts seeking to extract information from the wealth of ever-evolving online data. We encourage the broader research community to contribute to similar task-agnostic frameworks. Our framework is freely available at: github.com/Checktr/tadkg.


Merger and Acquisition Detection

Companies

Add Company

MICRO COMMERCIAL COMPONENTS CORP 3/5
PROSPERITY DIELECTRICS CO LTD 3/5
CENTRAL SEMICONDUCTOR CORP 4/5
OSRAM GMBH 4/5
ALTERA CORPORATION 4/5
EVERLIGHT ELECTRONICS CO LTD 3/5
DELTA ELECTRONICS INC 4/5
HARTING KGAA 2/5
MINI-SYSTEMS INC 3/5
MULTICOMP INC 3/5
TAIWAN SEMICONDUCTOR CO LTD 3/5
PAN JIT INTERNATIONAL INC 4/5

Webpages for "MICRO COMMERCIAL COMPONENTS CORP"

https://www.businesswire.com/news/home/20210813005132/en/Helbiz-Inc.-and-Greenvision-Acquisition-Corp.-Announce-Closing-of-Business-Combination-Helbiz-to-Trade-on-Nasdaq-Under-Symbol-%E2%80%9CHLBZ%E2%80%9D 4/5
https://www.reuters.com/article/eu-mergers-and-takeovers-idUSPRWP1420100705 NEW 4/5
https://spacenews.com/tag/mergers-and-acquisitions/ NEW 4/5
https://en.wikipedia.org/wiki/Microsemi 3/5
https://opentextbc.ca/principlesofeconomics/chapter/11-1-corporate-mergers/ NEW 3/5
https://www.computerhistory.org/siliconengine/companies/ 3/5

Scanned Webpage
https://www.reuters.com/article/eu-mergers-and-takeovers-idUSPRWP1420100705
Top Entities:
N (8), UL (4), V (3), PA (3), Bell Micro (2), JSA (2), Jacquet Metals (2), General Electric (1), Keolis Nordic AB (1), Keolis SA (1), Caisse de Depot et Placement (1), CDPDA (1), Busslink (1), Geodis (1), SNCF Participations (1), Giraud International (1), Univar Europe Holdings B (1), Eurochem Holding B (1), Otto (1), Intermadrid Holdings Cooperatief U (1), Autovam Holdings Cooperatief U (1), Eiser Infrastructure Ltd (1), Autovias de Peaje en Sombra, S (1), AVT Acquisition Corp (1), Lactalis (1), Warburg Pincus (1), Silver Lake (1),

appliance maker Whirlpool WHR.N to acquire the rights to the Privileg home appliances brand from German mail order group Otto (notified June 2/deadline July 7) JULY 9 -- Intermadrid Holdings Cooperatief U.A

Acquisition - Confidence: 5/5
and Spanish hob concession holder Autovam Holdings Cooperatief U.A., controlled by British private equity investor Eiser Infrastructure Ltd, acquire 49 percent of two newly set up Spanish motorway concession holders Autovias de Peaje en Sombra, S.L

Acquisition - Confidence: 5/5
IT distributor Bell Micro Inc BELM.O via a merger of its subsidiary AVT Acquisition Corp and Bell Micro (approved July 2) NEW LISTINGS: -- French dairy group Lactalis to acquire the dairy business of Spanish food company Ebro Puleva EVA.MC (notified July 2/deadline Aug

Acquisition - Confidence: 5/5
9) EXTENSIONS AND OTHER CHANGES: None FIRST-STAGE REVIEWS BY DEADLINE: JULY 5 -- Private equity firms Warburg Pincus and Silver Lake to acquire financial data provider Interactive Data Corp IDC.N (notified May 31/deadline July 5/simplified) -- Spanish publisher Grupo Planeta to acquire a stake in Spanish media club Circulo de Lectores from German media group Bertelsmann AG [BERT.UL] (notified May 31/deadline July 5) JULY 6 -- Chinese carmaker Zhejiang Geely Holding Group, parent of Geely Automobile Holdings 0175.HK, to acquire Ford Motor F.N's Volvo car unit (notified June 1/deadline July 6/simplified) JULY 7 -- German drugmaker Merck KGaA MRCG.DE to acquire U.S

Merger - Confidence: 5/5
(SyC) (notified June 5/deadline July 9/simplified) JULY 14 -- Belgian holding company JSA, which has sole control of steel product distributor Jacquet Metals JACQ.PA, to acquire sole control of the new entity created by the merger between France's IMS International Metal Service ITMT.PA ITMT.PA and Jacquet Metals (notified May 26/deadline extended to July 14 from June 30 after JSA offered commitments) JULY 15 -- British Airways BAY.L and Spanish carrier Iberia IBLA.MC to merge (notified June 10/deadline July 15) -- U.S

Acquisition - Confidence: 4/5
electronic components distributor Avnet AVT.N to acquire U.S

Figure 8: Interface with Data for Mergers and Acquisitions for Companies from List Provided by Lytica

REFERENCES

[1] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. https://doi.org/10.48550/ARXIV.2004.05150

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1247–1250.

[3] Xiaojun Chen, Shengbin Jia, and Yang Xiang. 2020. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications* 141 (2020), 112948.

[4] Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. 2020. Knowledge graph completion: A review. *Ieee Access* 8 (2020), 192435–192456.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/ARXIV.1810.04805

[6] Xin Luna Dong. 2018. Challenges and innovations in building a product knowledge graph. In *Proceedings of the 24th ACM SIGKDD International conference on knowledge discovery & data mining*. 2869–2869.

[7] Venkat Gudivada, Amy Apon, and Junhua Ding. 2017. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software* 10, 1 (2017), 1–20.

[8] Kevin A Hallgren. 2012. Computing inter-rater reliability for observational data: an overview and tutorial. *Tutorials in quantitative methods for psychology* 8, 1 (2012), 23.

[9] Dwayne Hipp. 2005. About SQLite. https://www.sqlite.org/about.html

[10] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 4 (2020), 1234–1240.

[11] Alex Luscombe, Kevin Dick, and Kevin Walby. 2022. Algorithmic thinking in the public interest: navigating technical, legal, and ethical hurdles to web scraping in the social sciences. *Quality & Quantity* 56, 3 (2022), 1023–1044.

[12] Lytica. 2022. About. https://www.lytica.com/about/

[13] Mergr. 2022. The Mission. https://mergr.com/about

[14] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[15] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *Commun. ACM* 61, 5 (2018), 103–115.

[16] Nv7. 2020. Googlesearch-python. https://pypi.org/project/googlesearch-python/

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. https://doi.org/10.48550/ARXIV.1912.01703

[18] Leonard Richardson. 2022. Beautiful Soup. https://www.crummy.com/software/BeautifulSoup/

[19] Amit Singhal. 2012. Introducing the knowledge graph: things, not strings. *Official google blog* 5 (2012), 16.

[20] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago: A large ontology from wikipedia and wordnet. *Journal of Web Semantics* 6, 3 (2008), 203–217.

[21] Jian Xu, Sunkyu Kim, Min Song, Minbyul Jeong, Donghyeon Kim, Jaewoo Kang, Justin F Rousseau, Xin Li, Weijia Xu, Vetle I Torvik, et al. 2020. Building a PubMed knowledge graph. *Scientific data* 7, 1 (2020), 1–15.

[22] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. https://doi.org/10.48550/ARXIV.1912.08777