Understanding Customer Requirements:
an Enterprise Knowledge Graph Approach

Basel Shbita[1*], Anna Lisa Gentile[2], Pengyuan Li[2], Chad DeLuca[2], and Guang-Jie Ren[2]

[1] University of Southern California, Los Angeles, CA, USA
and Information Sciences Institute, Marina del Rey, CA, USA
shbita@usc.edu
[2] IBM Research Almaden, San Jose, CA, USA
{annalisa.gentile, pengyuan}@ibm.com {delucac, gren}@us.ibm.com

Abstract. Understanding customers demands and needs is one of the keys to success for large enterprises. Customers come to a large enterprise with a set of requirements and finding a mapping between the needs they are expressing and the scale of available products and services within the enterprise is a complex task. Formalizing the two sides of interaction - the requests and the offerings - is a way to achieve the matching. Enterprise Knowledge Graphs (EKG) are an effective method to represent enterprise information in ways that can be more easily interpreted by both humans and machines. In this work, we propose a solution to identify customer requirements from free text to represent them in terms of an EKG. We demonstrate the validity of the approach by matching customer requirements to their appropriate business units, using a dataset of historical requirement-offering records in IBM spanning over 10 years.

1 Introduction

Capturing domain knowledge is a critical task in many domains and applications - and especially so when dealing with capturing information about large organizations. In a large enterprise, the information landscape of assets, skills, intellectual properties, and customer requirements is very complicated, extremely dynamic, and often under-specified - and this is especially true for a company like IBM, that provides information technology solutions and consulting services spanning from AI, analytics, security, cloud, supply chain etc. to a vast spectrum of industries including banking, finance, education, energy, healthcare, government, travel and transportation to mention a few. In many cases, the Subject Matter Experts (SMEs) are the gatekeepers for matching customer needs with the multitude of the company's offerings - and they are perceived to hold the required knowledge to steer and develop business opportunities for the company. A way to alleviate the issue is making sure that information is properly formalized, shared, and accessible within the organization. Knowledge Graphs (KG) are a

* This work was conducted during Summer Internship at IBM Research Almaden.

popular way to represent information in a way that can be easily interpreted by both humans and machines. In fact, for more than a decade many organizations adopted KGs - oftentimes referred to as Enterprise KG (EKG) - to lay the foundation of next-generation enterprise data and metadata management, search, recommendation, analytics, intelligent agents, and more [33, 60].

Nonetheless generating and maintaining a complete EKG that harmonizes all internal knowledge as well as data gathered from the vastness of open KGs on the Web can be expensive and difficult to achieve, for many reasons. Both internal organizational data and external customer requirements can be highly heterogeneous, cryptic, short, highly technically written, and only understood by a few. Those who actually have the expertise, experience, and procedural knowledge to understand such data often do not have the expertise to formalize it, nor the time to do so. External open knowledge available on the Web is vast, incomplete, imbalanced, highly distributed, and heterogeneous - and not easily accessible for SMEs without knowledge management skills. Without proper integration, the multitude of data repositories on the Web do not prove helpful in supporting decision-making systems.

In this work we address the problem of bootstrapping, populating, and maintaining an EKG - or a portion of it - relying on three main building blocks. Specifically we (i) extract information from internal unstructured data in the organization; (ii) use Open Knowledge gathered from the Web to augment the initial data either directly or by bootstrapping off-the-shelf weakly supervised data augmentation algorithms; and finally (iii) we rely on domain expertise to structure all acquired information into an EKG. Our proposal is a human-assisted pipeline that generates augmented representations of organizational concepts. Our humans-in-the-loop are SMEs internal to the organization who are familiar with the assets and can understand customer needs - and can help shape their semantic representation. The SMEs are involved in the selection of external knowledge to be included in the EKG, or to be used to bootstrap the training of the system. This approach lets us get high-quality, up-to-date insights from undersized, emerging corpora.

The main contribution of this work is a showcase of how the orchestration of NLP techniques, the usage of semantic resources, and human expertise can deliver industrial impact. We combine: (i) a method to tap into the vastness of open KG from the Web and retrieve, select, and distill instrumental knowledge to build and expand the EKG of interest; (ii) a method to extract structured information from unstructured technical text; and finally (iii) a user-driven mechanism to encapsulate the extracted knowledge within the existing EKG, and effectively expand it and populate it.

One of the most valuable uses of the constructed EKG is matching customer requirements to the available offerings in the organization. Specifically in this work we employ the extracted EKG to train automatic methods to help sales experts identify the correct business units and offerings to answer specific customer requirements. By using historical customer deal data within the organization, we show that our methodology can automatically retrieve the best business units

and offerings that satisfy a customer requirement (compared against historical data). Specifically, we designed an in vitro classification task, using curated ground truth from 10 years of historical sales deals and show that by using the fully constructed EKG we improve the F_1 score of the business-units multi-classification task by as much as 4.1% when compared to a baseline method without the usage of the EKG, and by 1.3% when using the non-augmented version of the EKG.

2 Related Work

Capturing, representing, persisting, and sharing the vast amount of human knowledge in a way that is effectively accessible from both human and automated methods is a topic that attracted numerous research efforts and still presents many open challenges [55]. For many years Semantic Web technologies [34] have provided formal models to represent explicit semantics in a machine-interpretable form, and in the last decade Knowledge Graphs became the de facto standard to accumulate and convey knowledge of the real world, especially in industry settings [20]. As a representation of semantic relations between entities, KGs have proven to be particularly relevant for natural language processing (NLP) [42], both to inform the task of knowledge acquisition - i.e. using NLP to extract formal models from text [57] - or for the purpose of knowledge application - i.e. exploiting KGs to enhance NLP techniques and language models [6, 28, 54]. The recent survey paper by Schneider et al. [42] provides a detailed account of all recent works of KGs and NLP, classified under these two macro-classes. Our work falls under the knowledge acquisition category - where all techniques of information extraction and data integration are involved. At high level, the Information Extraction (IE) methods can be grouped in two categories: (i) Open IE - where the purpose is understanding the text - and (ii) targeted IE - where the goal is a focused extraction of domain specific information for the purpose of Knowledge Population [41].

Open IE can be considered a form of machine reading [2, 13, 36], where the target knowledge schema is not known and extractions are represented in the form of surface subject-relation-object triples. It is worth noting that there are many solutions for text understanding that do not extract human understandable facts, but rather use statistical information to model the language in text [10, 25, 37, 61], and represent the information with numerical vectors that capture statistical patterns of words - with words embeddings [1] being a popular choice. While these data-driven models are impressive, they do not represent semantics explicitly - therefore data is unintelligible for a human - and they have no mechanism to logically reason about the captured knowledge. While we use some of these methodologies in our proposed pipeline, it is merely as a preprocessing step and we do not make specific contributions or claim novelties in this direction, therefore discussing them in detail is out of the scope of this work. On the other hand, some Open IE methods - more in line with our proposed approach - represent extracted information as a Knowledge Graph, including

FRED [14], SHELDON [38], ClausIE [9], MinIE [15] and other similar methods [16, 18]. The main difference with our proposed approach is that these methods do not assume domain specific target ontologies, therefore the extracted content can be way more than needed for the specific population task, thus requiring additional effort for the selection of relevant information.

Our work is similar to targeted IE methods, which collect the extracted information in formal models, i.e. ontology population from text [35, 56]. Some of the earliest approaches for ontology population from text are based on pattern matching, string similarity functions, and external glossaries and knowledge bases [52, 51, 4, 26, 44]. Others exploit vector-feature similarity using terms and n-grams as features [5, 47, 17]. More recent works rely on: (i) machine learning, with standard NLP features extracted from text [24], or more sophisticated features such as type relational phrases [31], or type correlation based on co-occurring entities [39]; (ii) graph embedding models [62, 59, 40] and (iii) deep learning models [12, 45, 46, 58, 30].

There exist many established initiatives to foster research on targeted IE, such as the Knowledge Base Population task at TAC,[3] the TREC Knowledge Base Acceleration track,[4] and the Open Knowledge Extraction Challenge [32]. In these initiatives, systems are compared on the basis of recognizing individuals belonging to a few selected ontology classes, spanning from the common Person, Place and Organization [48], to more specific classes such as Facility, Weapon, Vehicle [11], Role [32] or Drug [43], among others. All these works tackle the KG completion in terms of "missing instances", i.e. finding entities belonging to pre-defined given classes. In the same direction, there is a plethora of tools for automatically detecting named entities in free text and aligning them to a predefined knowledge base, i.e., Spotlight [27], X-Lisa [63], Babelfy [29], Wikifier [3]. However, all these tools are able to identify only instances that already exist in a knowledge base, but do not cover the case of out-of-knowledge entities - which is a common case for very technical and niche domains.

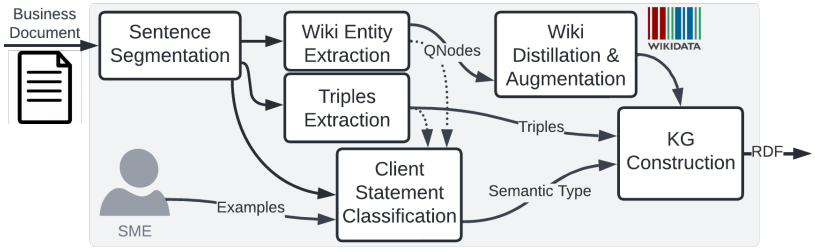

Fig. 1. EKG construction pipeline.

[3] http://www.nist.gov/tac/2015/KBP
[4] http://trec-kba.org/

3 Building the EKG

The problem we address can be classified as a data and knowledge acquisition and integration task. Formally, we wish to populate and augment the Enterprise Knowledge Graph (EKG) by automatically integrating specialized and complementary knowledge coming from heterogeneous sources: proprietary messy unstructured data in the form of textual documents describing client requirements, open linked data on the Web (Open KGs), and non-formalized SME knowledge in the business domain.

Our proposed pipeline for the construction and enhancement of the EKG consists of several steps and components, as illustrated in Figure 1. Each of the components is designed to address the data extraction and knowledge integration step in each of the data sources, then fuse them together to create an effective EKG. We firstly extract initial triples from the unstructured proprietary data (Section 3.2); then distill relevant knowledge from open KGs from the Web (Section 3.3); and finally structure all extracted information into an EKG with a user-driven method (Section 3.4).

3.1 Enterprise Knowledge Graph (EKG)

An Enterprise Knowledge Graph (EKG) is simply a KG of enterprise data with the goal of powering certain capabilities that are relevant to the business. The structure and content of the graph pertains to certain assets, clients, their requirements, and other relevant entities and offerings in the business domain.

In our problem setting, a manually-curated EKG is provided. The EKG contains triple statements about IBM business offerings/assets, organized in a hierarchical structure that reflects the organizational grouping of assets within business units and various topical verticals. The initial EKG used in this work contains 422 classes and 827 individuals.

In its initial form, the graph does not capture any knowledge about clients, their needs, or how these may relate to any of the pre-existing assets in the graph. Our task is: given client requirements in textual documents, we want to extract and represent client needs from text in a form that can be integrated and connected to the current EKG. This task requires materializing stakeholders, their needs, assets, and the relationships between all of the above within the text. While the EKG already contains information about assets, client needs are completely unknown to the EKG and they are expressed in text with a vocabulary that often has little or no intersection with the exiting EKG concepts. In order to tackle this problem, we enrich the extracted triples (both entities and relations) with additional knowledge existing on the open Web and benefit from prior knowledge provided by SMEs in the form of predefined examples of client statement types that are used to structure the final EKG. This allows us to capture corpus-specific information (both structural and semantic) and adapt to different scenarios effortlessly.

3.2 Extracting Triples from Proprietary Data: an NLP Approach

Business client requirement documents manifest as text snippets and include a diverse set of client-specific vocabularies, which vary by industry, client, or individual. These documents contain relevant and irrelevant information, which may be hard to distinguish even for an SME. Each document might contain multiple requirements, sometime expressed in the same short paragraph. Therefore, we must provide practical tools to achieve a complete analysis, considering particular and general needs and managing requirements in a comprehensive manner.

In this work we used a selection of "client cases", typically redacted by curation teams that summarise the profile of the client company, what kind of problem they are trying to address (the requirements), and which product(s) or service(s) they purchased from IBM that helped solved their problem. For the purposes of illustration, we will use the client case text shown in Listing 1.1 for all the examples.

```
<COMPANY NAME> uses a Microsoft Distributed File System to enable employees
to access data and files residing on different servers, usually to create
financial reports. Previously, <COMPANY NAME> was supporting its Microsoft
Distributed File System with older IBM Power 570 servers. Although the
incumbent technology had performed satisfactorily, the servers were nearing
the limits of their processing capacity and were starting to generate high
maintenance costs. In addition, <COMPANY NAME> recognized the opportunity
to increase storage capacity and streamline administration tasks within its
storage area network (SAN). Thus, <COMPANY NAME> sought a reliable IT provider
to help it refresh its server and storage technology and boost the performance
of its Microsoft Distributed File System.
```

Listing 1.1. An example of a short client case text.

We design a custom pipeline to perform triple extraction from technical text, based on linguistic parsing, statistical processing, and custom heuristics (Figure 1). The first step is sentence segmentation, which is the process of identifying different sentences in a text paragraph. We perform the triple extraction step for each individual sentence, i.e. we identify the pairs of concepts (entities) and the relation between them, in the form of <concept> <relation> <concept> (example in Figure 3).

The extraction is driven by (i) the part-of-speech (POS) annotation tags and (ii) the dependency tree of the sentence. POS tagging is the process of identifying the grammatical roles that explain how a particular word is used in a sentence (e.g., noun, adjective, verb). Dependency parsing is the process of extracting the dependency tree that represents the grammatical structure of the sentence (e.g. finding the subject and its relation to other words). An example of the extraction process is depicted in Figure 2, where underneath each word in the sentence we denote its POS tag and over each edge we indicate the grammatical relation between the words. Both of these extraction steps are implemented using off-the-shelf tools. Specifically we rely on components from *SpaCy* [21], a comprehensive Natural Language Processing (NLP) library.

Once each sentence is annotated we use a rule-based algorithm to extract the relations in the form of individual triples. Specifically, we build on the hypothesis

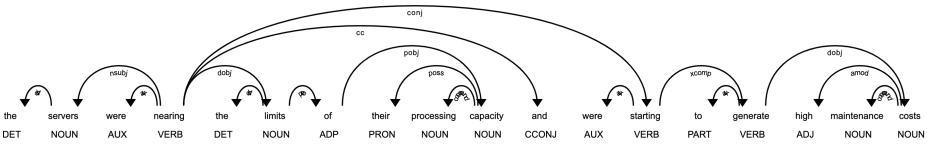

Fig. 2. Dependency parse tree of an example sentence. POS tags are shown below each word. The grammatical relation between words is denoted over the edges.

that the main relation, i.e. the predicate of the triple, is the main verb in the sentence. We identify this verb first, and we concatenate any auxiliary, preposition, and negation words when present. This becomes the predicate of our triple. Starting from the detected predicate ("root" verb) we traverse backwards over the dependency tree to identify the candidate subject. Any compound, prefix, or modifier words preceding the candidate subject will be concatenated to it, with the exception of words with a POS tag that is neither a noun or an adjective. We run the same process forward (following compound elements) to determine the candidate object as well. This heuristic is not guaranteed to be optimal, yet it provides a sufficient solution for reaching a good, yet noisy, extractor that will steer additional components in our pipeline. As an example, consider the sentence shown in Figure 2. In this case our technique will yield the verb "`were nearing`" as the predicate/relation. The subject/head is the company itself (employs the anonymized term "`<COMPANY NAME>`" to ensure confidentiality) and the object/tail is "`high maintenance costs`".

The full document (including the sentence shown in Figure 2) and the set of extracted triples are shown in Figure 3 - note that we use the company who's request belongs to as the subject/head of all the predicate/objects, as we want to create a simplified customer-centric representation. Figure 3 also shows in red boxes entities that have been externally recognized as entities in Wikidata [53] - we extract those using the *Wikifier* [3] Named Entity Recognition (NER) service. This extraction is performed passing the full text snippet to Wikifier, completely in parallel to the triple extraction process. Full details on the Wikidata based augmentation are provided in Section 3.3.

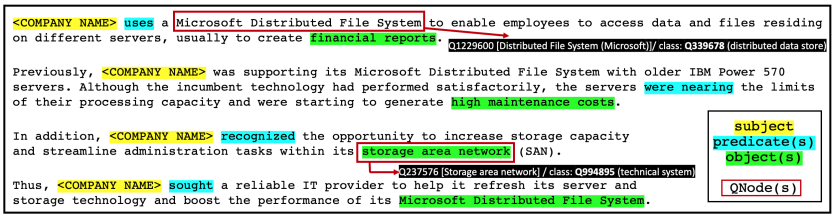

Fig. 3. Example document and its corresponding extracted triples (subjects are highlighted in yellow, predicates are highlighted in blue, and objects are highlighted in green), and linked QNodes (wrapped in a red, linked to the entity highlighted in black).

3.3 Open KG from the Web: a Distillery Approach

The semantic Web and the Linked Open Data (LOD) cloud[5] are an invaluable source of knowledge, with a multitude of semantically connected datasets to exploit. For the scope of this work, we selected Wikidata [53] to perform our data augmentation process. Wikidata is a free, collaborative, fast-growing KG and importantly acts as a hub in the LOD cloud, in the sense that contains numerous incoming and outgoing links to external KGs and ontologies. These sources can also be exploited for further enrichment of the data and can be used for future approaches to this task. Moreover, there are plentiful state of the art tools to bootstrap the linkage from text to Wikidata - in fact we use the *Wikifier* [3] Named Entity Recognition (NER) service in our augmentation pipeline.

The sheer size of Wikidata makes it difficult for an SME to effectively tap into it. For this reason we design a KG distillation algorithm that: (i) constructs a salient subgraph of Wikidata entities - referred in Wikidata as QNodes - that are relevant to the extracted triples at hand (resulting from the process described in Section 3.2) and (ii) leverages the Wikidata taxonomy to integrate essential relations between the QNodes in the distilled subgraph.

Figure 3 depicts an example document, the result of the triple extraction, and the corresponding Wikidata entities. In the distillation process we were able to connect the phrase (and object) "`storage area network`" in the document to a corresponding entry in Wikidata. We found out that "`Storage area network`" corresponds to QNode Q237576 and is an instance of the concept "`technical system`" (Q994895), and we also retrieved all the corresponding successor concepts from Wikidata (as seen in Figure 4).

To perform the distillation, we use the Wikidata SPARQL query service.[6] SPARQL is query language [7] and is one of the most powerful and most widely used services for accessing Linked Data. The core component of our KG distillation process is essentially a SPARQL query: the Subgraph Retrieval Query (SRQ). The SRQ query (depicted in Listing 1.2) starts with a target entity/-class that has been previously recognized by the Wikifier service; it retrieves all QNodes that are either direct instances of the target entity/class or of any of its sub-classes, recursively.

```
1 SELECT ?c ?cLabel WHERE {
2   wd:Q994895 wdt:P31?/wdt:P279* ?c .
3   SERVICE wikibase:label {
4     bd:serviceParam wikibase:language "en". } }
```

Listing 1.2. *Subgraph Retrieval Query (SRQ)*: The query has one target entity - QNode (Q994895) - and retrieves all entities that are *instance of* (`P31`) or *subclass of* (`P279`) the target entity (Q994895). The subclass relation is applied recursively - using the `*` notation. Line 3-4 of the query simply retrieve a human-readable label for each resulting QNodes.

[5] LOD cloud: https://lod-cloud.net/
[6] Wikidata SPARQL query service: https://query.wikidata.org/

Algorithm 1: Distilled Wikidata graph construction algorithm

Data: a set $\mathcal{Q}$ of QNodes
Result: a directed acyclic graph $\mathcal{G}$ of QNodes
1 **foreach** $q \in \mathcal{Q}$ **do**
2 $\mathcal{G}$.add(q); // add all QNodes to graph
3 **while** *True* **do**
4 $\mathcal{L}$ = list of nodes in $\mathcal{G}$ with no outgoing P31/P279 edges;
5 $\mathcal{Q}^*$ = `getP31P279`($\mathcal{L}$) $\setminus \mathcal{Q}$;
6 **foreach** $q^* \in \mathcal{Q}^*$ **do**
7 $\mathcal{G}$.add($\bigcup_{q_j^* \in P31P279(q)} q \mapsto q_j^*$); // add new nodes & edges
8 **if** $|\mathcal{Q}^*| == 0$ **then**
9 break;
10 $\mathcal{Q} = \mathcal{Q}^* \bigcup \mathcal{Q}$

To handle the distillation task efficiently and avoid unnecessary calls to the SPARQL endpoint, we designed Algorithm 1. The algorithm performs the augmented subgraph construction. First, all recognized QNodes are inserted as nodes to the empty graph $\mathcal{G}$ (lines 1-2). Then, we retrieve *leaf nodes* - nodes without any outgoing edges of type P31/P279 - from the graph $\mathcal{G}$ to list $\mathcal{L}$ (line 4). The function `getP31P279` (line 5) acts as a wrapper function which triggers the SRQ query for each input QNode in $\mathcal{L}$. The variable $\mathcal{Q}^*$ will hold the retrieved predecessor nodes (with respect to P31/P279) from Wikidata, not including QNodes that already exist in the graph (by subtracting $\mathcal{Q}$ in the same line). Next, we insert all the newly retrieved nodes and edges into the graph. The iterative addition over the set (line 7) is required since there may be one-to-many relations for each QNode, as expected in Wikidata, so we must execute over a set of nodes. We enforce a terminating case (line 8) that happens when there is no growth in the graph, meaning no new QNodes are retrieved, i.e. they already exist in our graph. The set $\mathcal{Q}$ is updated at each iteration (line 10) to reduce the lookup in the next repetition.

The relations between the nodes in the resulting graph $\mathcal{G}$ carry a semantic meaning between the different QNodes (a node is semantically similar to its predecessors and its successors) and will play a critical role in the RDF generation. Examples for such distilled subgraphs, are shown as green nodes in Figure 4. As seen in the graph, the QNode Q994895, which was retrieved following the extraction of the object "`storage area network`", allowed the retrieval and addition of QNodes such as Q58778 `system` to the graph with the appropriate taxonomic relations from Wikidata.

3.4 Formalizing the Final EKG: a User-assisted Semantic Typing Approach

Semantic typing is a group of fundamental natural language understanding problems aiming to classify tokens (or objects) of interest into semantic categories. In the context of our problem, this task aims to introduce meaningful semantic

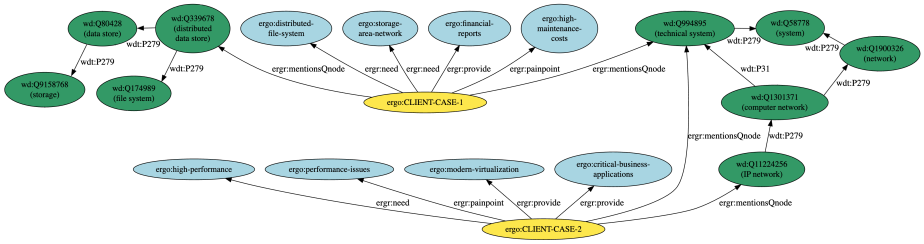

Fig. 4. A snippet of the EKG representing concepts extracted from client requirements with linkage to acquired QNodes. Client document nodes are shown in yellow. Extracted objects and concepts are shown in blue. QNodes are shown in green.

labels and types to the extracted objects in each triple, and allow users to narrow the focus of the resulting EKG.

Conventionally, SMEs can find it challenging to articulate their domain knowledge with a formalism such as RDF. Considering the amount of data the EKG holds, modeling the entire EKG becomes intractable and tedious and is often susceptible to human error. While one obvious solution is to perform automated semantic labeling, it is a difficult task in a narrow and technical domain. A middle course to a fully automated solution is to design a data-driven procedure that can leverage SME knowledge to guide the data understanding and produce semi-automated labeling services.

Specifically for our problem, we are trying to model client needs and requirements in the EKG. While recognizing the triples can be mostly achieved automatically, typing them often requires the definition of new concept classes. To this end, we provide random samples of the extraction (triples, augmented entities) to the SME and ask them to group the samples where they see fit. The grouping is guided by the purpose of each particular statement (and the subsequent augmented triple) in the client requirement text; in fact, client requests contain snippets with different intents: some might describe their background, some might describe what they are struggling with, etc. As an example, consider the text: "`[orgEntity] is one of the largest financial services ...`". This type of sentence is typically present in client requirement text, where a background of the client itself is provided. Similarly, the snippet: "`[orgEntity] relied on ... data warehouse environment to support ...`" is a common type of sentence describing current struggles of the client. We rely on the SMEs to freely identify these different intents and create a desired label for them. In fact, we give the SME a collection of approximately 20 sentences at a time from different documents, sampled randomly - each of which contains at least a single triple. We ask them to identify sentences that express a similar type of intent and provide us with a label (name) of what that intent might be. The SME can, for example, return us the set of two sentences "`[orgEntity] aims to be a personal bank in the digital age`" and "`[orgEntity] is one of the largest financial services companies in the United States`" and de-

clare that their intent is to "profile" an organization. Table 1 shows additional examples of different types of "Statement Types" that were defined with the help of SMEs. For the described four types, we engaged with one SME for a total annotation time of one hour. Technically speaking, we store this information (name of the intent, example sentences, etc.) in `YAML` files (Figure 5), effecetively translating the SME high-level reasoning into actionable training examples for the subsequent automated steps. A more technical SME can even introduce variance in the categorized sentences by introducing a "slot" that corresponds to entity types. In the example in Figure 5, the `orgEntity` slot has a partial list of values (of type `org`) to provide structure and flexibility in the general classification method.

Table 1. Client statement semantic types, their descriptions, and examples.

Statement type	Description	Example
`profileStatement`	General attributes about the entity	*"is one of the largest financial services..."*
`attemptStatement`	Attempt or historical actions the entity has taken	*"relied on... data warehouse env. to support..."*
`needStatement`	Needs or desired outcomes the entity requires	*"wanted to build a system that..."*
`painPointStatement`	Pain points or obstacles the entity finds critical	*"struggled to process large volumes..."*

This process results in a categorical classification model that is later used to automatically translate the rest of the data into a formal EKG representation. Specifically the classification model id implemented using *Snips-NLU* [8], and trained with data generated by the SMEs. *Snips-NLU* uses a simple regular expression matching as a first stop to match against statements from training and logistic regression if there is no direct match. For logistic regression, TF-IDF and co-occurrence features are extracted from the text, and a classifier is trained with simple stochastic gradient descent. It is easy to add new training examples and custom statements by configuring `YAML` files similar to the discussed example (Figure 5).

```yaml
# org entity
---
type: entity
name: org
automatically_extensible: yes
values:
  - 3M Co.
  - [HP, Hewlett-Packard]
  - IBM
  - Intel
```

```yaml
# profileStatement intent
---
type: intent
name: profileStatement
slots:
  - name: orgEntity
    entity: org
utterances:
  - "[orgEntity] aims to be a personal bank in the digital age"
  - "[orgEntity] is one of the largest financial services companies in the United States"
```

Fig. 5. Excerpts from a `YAML` file used to train the semantic typing engine.

The training process can be iterated several times: the SME initially only identifies a small set of examples, but is able to repeat the learning procedure until they consider the labeling performance satisfactory, thus incrementally improving the accuracy of "intent" representation. The modularity of the procedure

allows SMEs to explore and experiment with different categorizations, slots, and filters as they see fit.

Finally, the extracted knowledge graph is materialized in an RDF format. In our implementation, the `RDFLib`[7] is used to construct the graph triples and convert the data to a semantic knowledge graph. Figure 4 shows an excerpt of the resulting knowledge graph. The yellow instances (i.e., `CLIENT-CASE-X`) represent two example client documents. Each one is the subject/head of the extracted triple, where their objects/tails are client statements (blue nodes) of some type - e.g. `ClientPainpoint` and `ClientNeed`, respectively matching the relation type leading to that object, i.e. `ergr:painpoint` and `ergr:need`). The rest of the instances are in green, representing the QNodes we acquired from Wikidata and the distillation process.

4 Evaluation and Discussion

The purpose of our experiment is to evaluate how our proposed method and pipeline impacts downstream tasks and the usefulness of the constructed EKG. Specifically we choose one of our downstream tasks - a multi-label classification [50] of customer requirement texts.

Dataset. The experimental dataset comprises 10 years of historical data of customer requirements and the corresponding sold assets and services. Specifically, the dataset contains 24,180 customer stories documents, each of them with an assigned set of business-unit labels - which indicate which business units within the company provided products/services to answer each specific customer requirement. A sample of some of the textual data can be explored on the "IBM Customer Stories: As told by our Customers" blog[8] where short articles classified by Topics and Industries describe successful collaborations between IBM and several customers. These stories are clearly redacted for broader audience, and can be richer and better elaborated that the raw data we use in our experiments, but they can give an idea of how use cases are described.[9]

The average length of each requirement document is 127 words, with the longest document containing 1,213 words, the shortest one 3 words (very short documents are discarded as invalid, and are typically a copy of the title).

The business-unit labels assigned to each document vary from 1 to a maximum of 9, with an average of 2.44 labels per document, with a total number of possible distinct labels of 13. These act as ground truth for the task and are used as target for our experiments.

Multi-label Classification Task. Our experimental task consists of classifying an unseen document (unstructured textual data) with one or several matching

[7] https://rdflib.readthedocs.io/
[8] https://www.ibm.com/blogs/client-voices/
[9] https://www.ibm.com/blog/sustainability-begins-with-design/

business-units. After performing standard text pre-processing on the text (stemming, stop-words, removal etc.), we compare five different experimental settings, using: (i) text based features only; (ii) adding simple-KG features, (iii) adding SME-based augmented-KG features, (iv) adding Wikidata-based augmented-KG features, (v) adding all augmented-KG features (both SME and Wikidata).

Baseline. The first setting acts as baseline for our experiments. The constructed KG is not used at all in this case. Instead, we utilize a state-of-the-art document embedding model called *Doc2Vec* [23] to transform each document into a feature vector. *Doc2Vec* determines a low-dimensional representation (i.e., embedding) for a document: it learns a neural network using at each time one target word from the document and the words that surround it, as well as it uses a global contextual vector, associated with the document, as part of its predictive model. Although Transformer-based models have been outperforming other models in many NLP tasks [22], *Doc2Vec* is considered a simpler and faster model and can be a more useful choice for a medium-sized dataset [19].

In this setting, we use the embedding produced by the *Doc2Vec* model to train a One-vs-Rest (OvR) multi-label classification model that uses Support Vector Machines (SVM) with a linear kernel to classify input data into business units as a target. OvR constructs one classifier per class, which is trained to distinguish the samples in the single class from the samples in all remaining classes. This setting is denoted as "Doc2Vec" in Table 2.

KG Based Settings. To inject the KG in the classification model we employ *ComplEx* [49], a state-of-the-art KG embedding model. As described in Section 3, each client document becomes a graph (example in Figure 4), which contains Wikidata-concepts nodes, "document" nodes (representing the whole document) and all other object nodes extracted by the triple extraction process. This whole graph is fed to *ComplEx*, which produces embedding vectors per each node and edge in the graph - we fix the size of each produced vector to 50.

The objective of *ComplEx* is to learn a fixed low-dimensional representation (i.e., embedding) of entities and relations in the KG while preserving their semantic meaning. By representing each node as a combination of vectors and computing their dot product, we are able to capture relationships between nodes. The dot product is passed through a complex-valued function that allows learning vector representation for each node in the knowledge graph, including those representing client documents. The choice of *ComplEx* vs other available KG embeddings techniques is that it uses complex valued vectors that better capture anti-symmetric relationships.

We experiment with four different versions of the KG (results in Table 2): (i) without applying the augmentation step ("Doc2Vec + KG"); (ii) only integrating the semantic types acquired with the help of SME ("Doc2Vec + Augmented KG (SME)"); (iii) only augmenting the with the Wikidata entities ("Doc2Vec + Augmented KG (Wiki)"); (iv) or using the fully augmented version of the KG, wuth both SME and Wiki based augmentations ("Doc2Vec + Augmented KG (Full)").

Similarly to the "Doc2Vec" baseline, we use the OvR multi-label SVM classifier with the business unit labels as a target, but we concatenate the *Doc2Vec* embedding with the appropriate KG embedding.
In each experimental setting, we utilized a 100-dimensional vector representation for training and testing the models. Having the same vector size of 100 for the embedding space across all settings was enforced to exclude that the benefit could be due to a bigger embedding space rather than the type of captured content. To ensure fairness in the information capacity between the different KG settings, we concatenated a 50-dimensional vector obtained from *Doc2Vec* with a 50-dimensional vector from the KG *ComplEx* embedding process. We evaluated the models based on precision, recall, and F_1 scores, using ten-fold cross-validation that split the data into mutually exclusive subsets. In this approach, one subset was used as the testing set, and the remaining subsets were used for training the model.

Table 2. Results summary for the business units multi-classification task.

Method	Precision	Recall	F_1
Doc2Vec	0.730	0.590	0.653
Doc2Vec + KG	0.741	0.605	0.666
Doc2Vec + Augmented KG (SME)	0.754	0.608	0.673
Doc2Vec + Augmented KG (Wiki)	**0.761**	0.621	0.684
Doc2Vec + Augmented KG (Full)	**0.761**	**0.638**	**0.694**

The evaluation results in Table 2 show the precision, recall and F_1 (harmonic mean of precision and recall) for the task of assigning the appropriate business unit(s) to an unstructured and unlabeled client request input, using the baseline method and the four KG augmentation variants. Compared to the baseline, the "Doc2Vec + KG" setting obtains slightly better results in all measures, while the fully augmented pipeline ("Doc2Vec + Augmented KG (Full)") achieves the best results in terms of precision, recall, and F_1, outperforming any other permutation. In this setting, we capture both semantic and contextual information, incorporate expert knowledge (SME), and leverage a vast and diverse source of structured data (Wikidata). The precision of 0.761, recall of 0.638, and F_1 score of 0.694 demonstrate that this model can achieve a balance between correctly identifying positive cases and minimizing false positives and false negatives, resulting in improved overall performance compared to a baseline model that relies only on text features.

Technical specifications and further considerations. For the "Doc2Vec + KG" settings the total number of the generated triples was 150,022. The embeddings were computed in a matter of minutes using a workstation powered by NVIDIA GeForce RTX 2080 Ti GPU and Intel i7 CPU (GPU has 4352 cores and 11 GB DDR6 memory).
For the other augmentation settings, it is worth mentioning the importance of the size of graph $\mathcal{G}$ (Algorithm 1). Clearly, we could use the entire RDF dump of Wikidata to incorporate the linkage information, but this would result in a

massive graph with a big number of QNodes, many of which not be useful for our task. This would generate an intractable number of computations when generating the KG embeddings - and many of those would be purposeless. This is where the distilled Wikidata graph comes into play. In this scenario, Algorithm 1 enriched our originally identified 3,729 QNodes to a total of 4,842 QNodes (connected within the directed graph $\mathcal{G}$), in 15 iterations (SPARQL calls - each with a response time averaging in 3-10 seconds) - again, in a matter of minutes. Following the semantic type assignments resulting from the SME inputs presented in Table 1, we retained statement of types: `attemptStatement`, `needStatement`, `painPointStatement` - and excluded statements of type `profileStatement` which simply describe the customer profile. It resulted in a total of 147,992 triples (including the materialized QNodes) - with 474 unique `needs`, 390 unique `painPoints`, and 123 unique `attempts` in the final EKG.

5 Conclusion and Future Work

In this work we introduce a method for constructing, modeling, and augmenting an Enterprise Knowledge Graph using (i) unstructured textual data from a collection of business requirement documents, (ii) open Knowledge Graphs from the Web - specifically Wikidata, and (iii) input from Subject Matter Experts. We evaluate our method using a dataset of historical records spanning over 10 years, capturing customer requests and products/services that have been provided to answer each specific customer requirement. We construct a graph from this data and quantify its the effect on informing a downstream task: classifying customer requests to one or more business units that can answer each specific customer need. The EKG improves the F_1 score of the classification task by as much as 4.1%. The aim of this work is for the augmented EKG to help the sales people navigating and browsing the multitude of company offerings along different business units, divisions and third party software offerings that we offer to our clients. Sometimes it can be difficult to understand what some of the assets accomplish (their descriptions can be highly technical). Having relations in the graph between a product and e.g. the pain-point extracted from client-stories (such as "`struggled to process large volumes of orders`") can be beneficial to understand their scope.

We foresee multiple directions for our future work. we plan to test with a wider variety of fine grained targets for classification - e.g. item recommendation. We will also explore multi-lingual support: matching requests in other languages to our business offerings - for which we always have a representation in English. We envisage using the EKG for various additional tasks, including link prediction, assessing product similarities, understanding patterns etc. Finally, we plan to extend our approach by leveraging additional textual knowledge from open KBs to enrich each client document - this can provide additional context and insights for the embedding process.

References

1. Almeida, F., Xexéo, G.: Word embeddings: A survey. arXiv preprint arXiv:1901.09069 (2019)
2. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the web. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence. pp. 2670–2676. IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2007), http://dl.acm.org/citation.cfm?id=1625275.1625705
3. Brank, J., Leban, G., Grobelnik, M.: Annotating documents with relevant wikipedia concepts. Proceedings of SiKDD **472** (2017)
4. Castano, S., Peraldi, I.S.E., Ferrara, A., Karkaletsis, V., Kaya, A., Möller, R., Montanelli, S., Petasis, G., Wessel, M.: Multimedia interpretation for dynamic ontology evolution. Journal of Logic and Computation **19**(5), 859–897 (2008)
5. Cimiano, P., Völker, J.: Towards large-scale, open-domain and ontology-based named entity classification. In: RANLP (2005)
6. Colon-Hernandez, P., Havasi, C., Alonso, J., Huggins, M., Breazeal, C.: Combining pre-trained language models and structured knowledge (2021), http://arxiv.org/abs/2101.12294
7. Consortium, W.W.W., et al.: Sparql 1.1 overview (2013)
8. Coucke, A., Saade, A., Ball, A., Bluche, T., Caulier, A., Leroy, D., Doumouro, C., Gisselbrecht, T., Caltagirone, F., Lavril, T., et al.: Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. arXiv preprint arXiv:1805.10190 pp. 12–16 (2018)
9. Del Corro, L., Gemulla, R.: ClausIE: Clause-based open information extraction. WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web (i), 355–365 (2013)
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
11. Doddington, G.R., Mitchell, A., Przybocki, M.A., Ramshaw, L.A., Strassel, S., Weischedel, R.M.: The automatic content extraction (ace) program-tasks, data, and evaluation. In: LREC (2004)
12. Dong, L., Wei, F., Sun, H., Zhou, M., Xu, K.: A hybrid neural model for type classification of entity mentions. In: IJCAI. pp. 1243–1249 (2015)
13. Etzioni, O., Fader, A., Christensen, J., Soderland, S., Mausam, M.: Open information extraction: The second generation. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One. pp. 3–10. IJCAI'11 (2011), http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-012
14. Gangemi, A., Presutti, V., Reforgiato Recupero, D., Nuzzolese, A.G., Draicchio, F., Mongiovì, M.: Semantic web machine reading with FRED. Semantic Web (Preprint), 1–21 (2016)
15. Gashteovski, K., Gemulla, R., del Corro, L.: MinIE: Minimizing facts in open information extraction. EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings pp. 2630–2640 (2017). https://doi.org/10.18653/v1/d17-1278
16. Gerber, D., Hellmann, S., Bühmann, L., Soru, T., Usbeck, R., Ngonga Ngomo, A.C.: Real-time rdf extraction from unstructured data streams. In: International semantic web conference. pp. 135–150. Springer (2013)

17. Giuliano, C., Gliozzo, A.: Instance-based ontology population exploiting named-entity substitution. In: ACL 2008. pp. 265–272. ACL (2008)
18. Hamoudi, Y., Comebize, T.: Extracting rdf triples using the stanford parser (2016)
19. Hoberg, G., Knoblock, C.A., Phillips, G., Pujara, J., Raschid, L., Qiu, J.: Filling the private firm void: Using representation learning to identify competitor relationships between businesses (2022)
20. Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G.D., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., Ngomo, A.C.N., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge graphs **54**(4) (2021), https://doi.org/10.1145/3447772
21. Honnibal, M., Montani, I.: spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing (2017)
22. Kalyan, K.S., Rajasekharan, A., Sangeetha, S.: Ammus: A survey of transformer-based pretrained models in natural language processing. arXiv preprint arXiv:2108.05542 (2021)
23. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International conference on machine learning. pp. 1188–1196. PMLR (2014)
24. Ling, X., Weld, D.S.: Fine-grained entity recognition. In: AAAI'12. pp. 94–100. AAAI Press (2012), http://dl.acm.org/citation.cfm?id=2900728.2900742
25. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
26. McDowell, L.K., Cafarella, M.: Ontology-driven, unsupervised instance population. Web Semantics: Science, Services and Agents on the World Wide Web **6**(3), 218–236 (2008)
27. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: Dbpedia spotlight: shedding light on the web of documents. In: Proceedings of the 7th international conference on semantic systems. pp. 1–8. ACM (2011)
28. Moiseev, F., Dong, Z., Alfonseca, E., Jaggi, M.: SKILL: Structured Knowledge Infusion for Large Language Models pp. 1581–1588 (2022). https://doi.org/10.18653/v1/2022.naacl-main.113
29. Moro, A., Raganato, A., Navigli, R.: Entity linking meets word sense disambiguation: a unified approach. Transactions of the Association for Computational Linguistics **2**, 231–244 (2014)
30. Murty, S., Verga, P., Vilnis, L., McCallum, A.: Finer grained entity typing with typenet. arXiv preprint arXiv:1711.05795 (2017)
31. Nakashole, N., Tylenda, T., Weikum, G.: Fine-grained semantic typing of emerging entities. In: ACL (1). pp. 1488–1497 (2013)
32. Nuzzolese, A.G., Gentile, A.L., Presutti, V., Gangemi, A., Garigliotti, D., Navigli, R.: Open knowledge extraction challenge. In: Semantic Web Evaluation Challenge. pp. 3–15. Springer International Publishing (2015)
33. Pan, J.Z., Vetere, G., Gomez-Perez, J.M., Wu, H.: Exploiting linked data and knowledge graphs in large organisations. Springer (2017)
34. Patel, A., Jain, S.: Present and future of semantic web technologies: a research statement. International Journal of Computers and Applications **43**(5), 413–422 (2021)
35. Paulheim, H.: Automatic Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods. SWJ **0**, 1–0 (2015). https://doi.org/10.3233/SW-160218
36. Presutti, V., Nuzzolese, A.G., Consoli, S., Gangemi, A., Reforgiato Recupero, D.: From hyperlinks to semantic web properties using open knowledge extraction. Semantic Web **7**(4), 351–378 (2016)

37. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. Tech. rep., OpenAI (2019)
38. Recupero, D.R., Nuzzolese, A.G., Consoli, S., Presutti, V., Peroni, S., Mongiovì, M.: Extracting knowledge from text using SHELDON, a semantic holistic framEwork for LinkeD ONtology data. WWW 2015 Companion - Proceedings of the 24th International Conference on World Wide Web pp. 235–238 (2015). https://doi.org/10.1145/2740908.2742842
39. Ren, X., He, W., Qu, M., Huang, L., Ji, H., Han, J.: Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In: Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP) (2016)
40. Ristoski, P., Faralli, S., Ponzetto, S.P., Paulheim, H.: Large-scale taxonomy induction using entity and word embeddings. Proceedings - 2017 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2017 pp. 81–87 (2017). https://doi.org/10.1145/3106426.3106465
41. Saggion, H., Funk, A., Maynard, D., Bontcheva, K.: Ontology-based information extraction for business intelligence. In: The Semantic Web, pp. 843–856. Springer (2007)
42. Schneider, P., Schopf, T., Vladika, J., Galkin, M., Simperl, E., Matthes, F.: A Decade of Knowledge Graphs in Natural Language Processing: A Survey (2022), http://arxiv.org/abs/2210.00105
43. Segura-Bedmar, I., Martínez, P., Herrero Zazo, M.: Semeval-2013 task 9 : Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). In: SemEval 2013. pp. 341–350. ACL (June 2013)
44. Shbita, B., Rajendran, A., Pujara, J., Knoblock, C.A.: Parsing, representing and transforming units of measure. In: Proceedings of the Conference on Modeling the World's Systems (2019)
45. Shimaoka, S., Stenetorp, P., Inui, K., Riedel, S.: An attentive neural architecture for fine-grained entity type classification. arXiv preprint arXiv:1604.05525 (2016)
46. Shimaoka, S., Stenetorp, P., Inui, K., Riedel, S.: Neural architectures for fine-grained entity type classification. arXiv preprint arXiv:1606.01341 (2016)
47. Tanev, H., Magnini, B.: Weakly supervised approaches for ontology population. Citeseer (2008)
48. Tjong Kim Sang, E.F., De Meulder, F.: Introduction to the conll-2003 shared task: Language-independent named entity recognition. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4. pp. 142–147. CONLL '03, Association for Computational Linguistics, Stroudsburg, PA, USA (2003)
49. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International conference on machine learning. pp. 2071–2080. PMLR (2016)
50. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. International Journal of Data Warehousing and Mining (IJDWM) **3**(3), 1–13 (2007)
51. Velardi, P., Faralli, S., Navigli, R.: Ontolearn reloaded: A graph-based algorithm for taxonomy induction. Computational Linguistics **39**(3), 665–707 (2013)
52. Velardi, P., Navigli, R., Cuchiarelli, A., Neri, R.: Evaluation of OntoLearn, a methodology for automatic learning of domain ontologies. Ontology Learning from Text: Methods, evaluation and applications **123**, 92 (2005)
53. Vrandecic, D., Krotzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM **57**(10), 78–85 (2014)

54. Wang, S., Zhao, R., Chen, X., Zheng, Y., Liu, B.: Enquire one's parent and child before decision: Fully exploit hierarchical structure for self-supervised taxonomy expansion. The Web Conference 2021 - Proceedings of the World Wide Web Conference, WWW 2021 pp. 3291–3304 (2021). https://doi.org/10.1145/3442381.3449948
55. Weikum, G., Dong, X.L., Razniewski, S., Suchanek, F.: Machine knowledge: Creation and curation of comprehensive knowledge bases. Foundations and Trends in Databases **10**(2-4), 108–490 (2021). https://doi.org/10.1561/1900000064
56. Weikum, G., Hoffart, J., Suchanek, F.: Ten Years of Knowledge Harvesting: Lessons and Challenges. Data Engineering **5**, 41–50 (2016)
57. Wong, W., Liu, W., Bennamoun, M.: Ontology learning from text: A look back and into the future. ACM Computing Surveys (CSUR) **44**(4), 1–36 (2012)
58. Yaghoobzadeh, Y., Adel, H., Schütze, H.: Noise mitigation for neural entity typing and relation extraction. arXiv preprint arXiv:1612.07495 (2016)
59. Yaghoobzadeh, Y., Schütze, H.: Corpus-level fine-grained entity typing using contextual information. arXiv preprint arXiv:1606.07901 (2016)
60. Yan, J., Wang, C., Cheng, W., Gao, M., Zhou, A.: A retrospective of knowledge graphs. Frontiers of Computer Science **12**(1), 55–74 (2018)
61. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. In: Advances in neural information processing systems. pp. 5753–5763 (2019)
62. Yogatama, D., Gillick, D., Lazic, N.: Embedding methods for fine grained entity type classification. In: ACL (2). pp. 291–296 (2015)
63. Zhang, L., Rettinger, A.: X-LiSA: cross-lingual semantic annotation. VLDB **7**(13), 1693–1696 (2014)