

Automatic Taxonomy Induction and Expansion

Nicolas Rodolfo Fauceglia, Alfio Gliozzo, Sarthak Dash,
Md Faisal Mahbub Chowdhury and Nandana Mihindukulasooriya

IBM Research AI, Yorktown Heights, NY, USA

{nicolas.fauceglia, nandana.m} @ibm.com

{gliozzo, sdash, mchowdh} @us.ibm.com

Abstract

The Knowledge Graph Induction Service (KGIS) is an end-to-end knowledge induction system. One of its main capabilities is to automatically induce taxonomies¹ from input documents using a hybrid approach that takes advantage of linguistic patterns, semantic web and neural networks. KGIS allows the user to semi-automatically curate and expand the induced taxonomy through a component called `smart spreadsheet` by exploiting distributional semantics. In this paper, we describe these taxonomy induction and expansion features of KGIS. A screencast video demonstrating the system is available in <https://ibm.box.com/v/emnlp-2019-demo>.

1 Introduction

Knowledge Graph Induction Service (KGIS) is an end-to-end knowledge graph (KG) induction system. Among other capabilities, it enables *automatic* taxonomy induction and *human-in-the-loop* curation. The output taxonomy representation can be used by downstream applications such as dialog systems and search engines.

The taxonomy is induced directly from the input documents by a combination of different approaches: one based on linguistic-patterns, another that leverages the semantic-web, and finally a novel neural network for cleaning and expanding taxonomies. The induced taxonomies are accessible through another component of the KGIS called `Smart Spreadsheet (SSS)`, which consists of an editable interactive tabular grid where the first row contains induced *types* (aka hypernyms), and each corresponding column contains its *instances* (aka hyponyms), henceforth, simply *types* and *instances*.

¹A *taxonomy* is a classification of things or concepts.

Using the SSS, the user can refine the automatically induced taxonomy, both by removing wrong *types* or *instances*, and by further expanding the *instances* of a particular *type*. This is accomplished by leveraging its main features: *auto-complete*², *type-suggestion*, and automatic population from *semantic web* and *distant supervision* using external Knowledge Graphs (KG). KGIS also allows taxonomies to be exported in RDF/OWL representation with the terms linked to Wikidata³ entities (Vrandečić, 2012).

We describe the use case in question in Section 2, followed by the system description for taxonomy induction component in Section 3. Then, in Section 4, we illustrate how the features of the SSS can be used for taxonomy curation and expansion. Finally, in Section 5 we conclude by discussing example outputs of the system.

2 Use Case Scenario




A typical KGIS user has a collection of documents that contain knowledge about a specific domain. The size of the collection makes human analysis or annotation impractical (slow and expensive). Therefore, the user ingests its collection into the KGIS, and fires a *KG Induction* job. Depending on the size of the collection, in minutes or a few hours the system provides the following artifacts: an annotated corpus, a terminology, a type embedding model, and different types of taxonomies that can be further refined.

²The option of completing terms by selecting from a suggested list of terms on the basis of the letters that has been already typed by the user.

³<https://www.wikidata.org>

KGIS - Home

Select an existing corpus

corpus name	comments	status	modified	created	
NeurIPS papers (1987 to 2017)	from WDS	KG induced	01/07/2019 - 18:12hs	01/07/2019 - 15:12hs	
it support (4k) and wikipedia (4.5K)	from WDS	KG induced	02/07/2019 - 16:19hs	02/07/2019 - 15:08hs	
medical corpus	from WDS	KG induced	22/07/2019 - 14:07hs	22/07/2019 - 14:01hs	

[Create new corpus](#)

Figure 1: KGIS home

3 System Description

3.1 Input processing and annotation

The input of the system is a collection of documents. The corpus creation process transforms different formats of inputs (txt, json, pdf, word, etc) into a standard document format. This is done through the *Document Conversion Service API* in IBM CloudTM 4.

After corpus creation, the system annotates terms (analyzing all noun phrases). This is done through the usage of *IBM Watson® NLU API*⁵. Next, all terms that have a frequency greater than a threshold (specified by the user during *KG Induction*, see figure 3) are combined to form the terminology.

3.2 Type Models

As part of the pipeline, the system builds a *type* model, which captures *type* similarity between two given domain terms (e.g. *SVM* and *Logistic Regression* are both of type *supervised learning algorithm*). To do so, the system trains a Word2Vec model (Mikolov et al. (2013)) using Gensim⁶ with a CBOW configuration of window size 2. The reason for this choice is that *type* similarity is captured by analyzing the local context around the terms. In addition, other input parameters for this model (e.g., embedding dimensions, min-frequency, learning rate, etc.), can be provided by the user while firing the *KG Induction*.

3.3 Taxonomy Induction

The Taxonomy Induction module takes the annotated corpus (see Section 3.1) and identifies *is-a* relations between pairs of terms. KGIS uses three

⁴<https://cloud.ibm.com/docs/cli>

⁵<https://cloud.ibm.com/apidocs/natural-language-understanding>

⁶<https://radimrehurek.com/gensim/>

different approaches for inducing shallow (*i.e.*, not hierarchical) taxonomies, as described in the following sections. The results of each approach are presented as an SSS where the users can manually validate the generated taxonomies and further curate them (see Section 4).

3.3.1 Pattern-Based Approach

For extracting *type-instance* relationships, *i.e.* *is-a* pairs, the system makes use of 24 lexico-syntactic patterns (e.g., “NP_y is a NP_x”), aka Hearst-like patterns (Hearst (1992)). Once the *is-a* pairs are extracted, the system applies the following pre-processing steps. First, the list of *is-a* pairs are represented as a graph by considering each pair as an *edge*, and the corresponding terms as nodes. The weight of each edge is the count of how often a pair has been extracted by pattern-matching.

The system excludes any cycle inside the graph; e.g. if (x, y) , (y, z) and (z, x) are present in the list of extracted *is-a* pairs, then all of them are discarded. KGIS also discards all edges that have a value lower than a frequency threshold f , which is specified by the user. KGIS checks all *types* and tries to identify potential proper nouns among them, using the following heuristic – a term x_1 would be considered as a proper noun if all the following three conditions hold: (i) it is not a substring of a *type* x_2 and vice-versa, (ii) x_2 is a *type* of x_1 , and (iii) x_2 belongs to a list of *types*⁷ that are known to have massive amount of proper noun instances. If a *type* is identified as a proper noun, KGIS discards all edges that link the term as *type* with any other term.

The next step of the pre-processing is to expand the list of pairs in the graph with likely (but not yet extracted) pairs of nested term *types* and super

⁷To be specific, “person”, “place”, “organization”, and “name”.

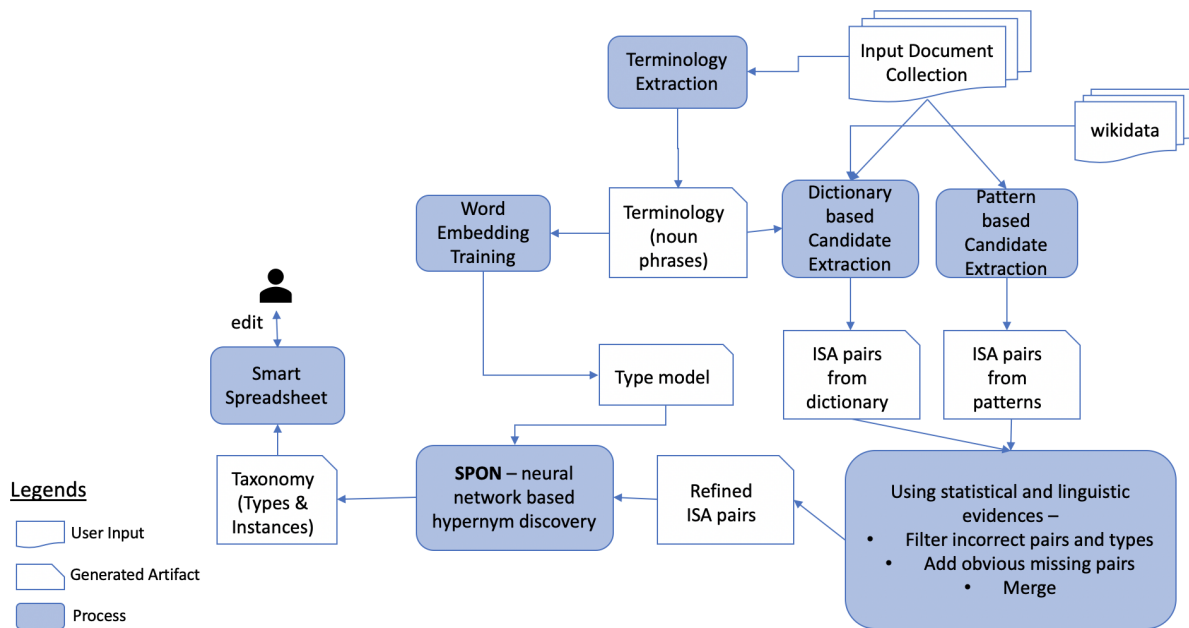


Figure 2: Taxonomy induction and expansion/curation workflow.

Corpus: **NeurIPS papers (1987 to 2017)**

reuse annotated docs:

reuse terminology:

reuse text for downstream:

annotator batch_size:

min count:

embeddings dim:

embeddings epochs:

embeddings initial learning rate:

embeddings force_train_from_scratch:

embeddings force_train_incremental:

Figure 3: KG induction

term *instances*. A nested term is a term that is part of a larger term (called super term).⁸ Often there are *types* that are nested terms in one or many of their *instances*. For example, the *type* “bank” is a nested term in *instances* “Capital One BankTM”, “Chase BankTM”, and “Bank of AmericaTM”.

KGIS builds a nested term search index

⁸For example, the terms “IBM” and “Corporation” are nested terms in their super term “IBM Corporation”.

(Chowdhury and Farrell, 2019) using the list of remaining *is-a* pairs. For every single word *type*⁹, the system identifies corresponding super terms from the index which are not already identified by the Hearst-like patterns as potential *instances*. If the *type* is the syntactic head word of one of these super terms, then KGIS adds that super term as a *type* in the list of *is-a* pairs.

The final list of pairs after all these steps is the output of the Pattern-Based approach.

3.3.2 Semantic Web Based Approach

Wikidata is a comprehensive cross-domain source of knowledge semantically represented using RDF and OWL. It contains a large amount of taxonomic information with relations such as *rdf:type* or *rdfs:subClassOf*. The semantic web based approach for candidate generation in KGIS consists on linking the terms identified in terminology extraction to Wikidata entities and discovering *is-a* relations between them using the background knowledge. Linking is done by approximate matching of the surface form of the term with the label and the aliases of the Wikidata candidate entity.

Once the initial list of *is-a* pairs is extracted, further filtering is performed to eliminate terms that might be erroneously linked. This is done by comparing the *type-instance* similarity and pairwise

⁹We avoid multi-word *types* as we found that exploitation of them results in significant amount of erroneous *is-a* pairs.

instance-instance similarity (using the Type model mentioned in Section 3.2) against a threshold that is provided as a parameter.

3.3.3 Neural Network Based Approach

Is-a relations are reflexive and transitive but not symmetric (Miller et al., 1990; Hearst, 1992). Using ideas from *order theory*, we can model these relations as *strict partial order* relations, i.e. a binary relation that is transitive, asymmetric and ir-reflexive.

We developed STRICT PARTIAL ORDER NETWORKS (SPON), a novel neural network architecture comprising of *non-negative* activations and *residual* connections designed to enforce strict partial order as a soft constraint. We use SPON as a component within the KGIS platform to model *is-a* relationships among pairs that have been generated from the approaches described in the previous Sections 3.3.1 and 3.3.2.

SPON works in three stages. In the first stage, it models the *is-a* relationships extracted in the previous steps and aims to provide a score to each individual *is-a* pair. In the second stage, it uses the same learned model in order to generate a *top-k* ranked list of *types* for every term present in the *terminology*. Finally, it ranks the accumulated *is-a* relationships generated in the previous steps according to certain *user-specified* criterion, and presents it back to the user.

Stage One. The purpose of this step is to score existing *is-a* relationships \mathcal{T} . This stage assumes that a true *is-a* relationship t is more likely to be inferred correctly, as opposed to a false relationship f when evaluated against a SPON model that has been trained using other *is-a* relationships i.e. $\mathcal{T} \setminus \{t, f\}$.

Following the assumption, the list of existing *is-a* relationships obtained via either Pattern based approaches or Semantic Web based approaches, are divided into *k-folds*. k independent SPON processes are then run in parallel, each process then trains upon $k-2$ folds, performs early stopping based on evaluations on the $(k-1)^{th}$ fold and finally generates scores for the k^{th} fold. The results for each fold are then concatenated together to generate the output \mathcal{O}_1 of Stage One.

Stage Two. The purpose of this step is to generate a ranked list of *types* for all the terms extracted in the Terminology extraction step (Section 3.1), but was not included in the list of *is-a* relationships

extracted using either Pattern based or Semantic Web based approaches.

Following similar approach as of Stage One, the list of existing *is-a* relationships are divided into *k-folds*. k independent SPON processes are then run in parallel, wherein each process trains upon $k-1$ folds, performs early stopping based on evaluations on the $(k-1)^{th}$ fold and finally generates a ranked list of *types* for all the terms in the Terminology.

Once all the SPON processes are over, we obtain k different ranked lists of *types* for each term in the Terminology. These ranked lists are then averaged to obtain a single ranked list of *types* per term. This generated output \mathcal{O}_2 then behaves as output for Stage Two.

Stage Three. Concatenating the results \mathcal{O}_1 and \mathcal{O}_2 from previous stages, we obtain an *extended* list of ranked *instances* for each *type*.

This stage, then works in two steps, in the first step all the *instances* for a given *type* whose score is less than a threshold θ are removed. In the second step, the *types* are then ranked by an average score of its *top m instances*.

Note that the parameters θ and m are entered by the user. The output of this step provides the final result for SPON component in KGIS.

4 Knowledge Curation using the Smart Spreadsheet (SSS)

The KGIS framework features a novel way of interacting with the induced knowledge called Smart Spreadsheet (SSS). The cells in an SSS correspond to the nodes in a KG ¹⁰. In addition, within an SSS, the first row is reserved for induced *types*, and other cells in each column contain *instances* of the corresponding *type* (i.e. the term in the 1st row of the column). Functionality wise, SSS provides: *auto-complete of term names*, for easily identifying terms matching an input text; *type-suggestion*, to help the user assign a *type* to a set of *instances*, and also *suggestion of similar or related terms* given existing ones.

Each time a new taxonomy is created (Section 3.3), it is also saved as an SSS so that the user can modify it to match their business needs. In the description that follows, the example snapshots following the functionality descriptions are taken from an automatically induced taxonomy,

¹⁰cells whose content does not match the *terminology* are painted *red*

Term	Predicted types
dicoumarol	<u>drug</u> , carbohydrate, acid, person, service, ...
Planck	person , <u>particle</u> , physics, <u>elementary particle</u> , service, ...
Belt Line relief	main road, infrastructure , transport infrastructure , expressway, way, ...
honesty	<u>virtue</u> , ideal, moral philosophy, philosophy, chastity, ...
shoe	footwear , shoe, footgear , overshoe, sandal, ...
ethanol	alcohol , <u>fuel</u> , person, fluid , resource, ...
ruby	<u>language</u> , <u>precious stone</u> , <u>person</u> , resource, stone , ...

Table 1: Examples of ranked predictions (from left-to-right) made by our system on a set of eight *randomly* selected test queries from SemEval 2018 English dataset. *Types* predicted by SPON that match the gold annotations are highlighted in **bold**, while we use underline for predictions that we judge to be correct but are missing in the gold standard expected *types*.

obtained by running the steps described in Section 3.3 on a corpus of scientific papers from the NeurIPS conference.

Firstly, given a list of few terms by the user supposedly belonging to a *type*, SSS has the ability to generate additional terms belonging to the same *type* using the following three options,

- **Populate from seeds.** The embeddings for the user-entered terms are averaged together to create a *centroid* vector; and a nearest neighbor algorithm is run across all the terms in the terminology to obtain additional terms belonging to the same *type*.
- **Populate from KG Type.** This option is only available when the user has found a matching *type* from the Semantic Web (using *type* suggestion). It allows the user to get the *instances* in the target KG are of this column’s *type* and are also present in the corpus’ terminology.
- **Populate from KG, Distant Supervision.** This option also requires a selected KG *type*, and it is a combination of the previous two techniques: it retrieves more *instances* of the given *type* from the KG, then it combines them with the ones already in the column (seeds) and applies the same ‘populate from seeds’ technique.

Figure 4b shows a snapshot of the KGIS system that demonstrates the `Populate from` feature. In this example, the user enters the name of a few algorithm names as *instances*, sets *algorithm* as the *type*, then selects a few cells and right-clicks to invoke the window containing the `Populate from` feature.

In addition, SSS provides an auto-complete feature, i.e. the option of completing terms wherein the list of displayed terms conform to the Terminology extraction step as discussed in Section 3.1. For example, in figure 4a as soon as the user types the term `spectral`, the system suggests meaningful possible terms to auto-complete based on the input corpus.

5 Conclusion and Future Work

Table 1 shows the result of applying our KGIS system on the English Domain corpus of the *SemEval 2018 Hypernym Detection* shared task. Two sets of four query terms (from test set input) are chosen at random. The first four terms have corresponding *is-a* pairs in the gold annotation provided by the task organizers, whereas the final four terms do not. The right hand column presents a ranked list of *types* for each query term.

Note that the correct *types* in the gold label test set are set in bold, whereas underlined terms are the *types* which we believe to be correct. This qualitative example, together with the academic benchmark results (not reported in this demo paper) together demonstrate that our system is the state of the art in discovering *is-a* pairs from text.

In conclusion, we have introduced the Knowledge Graph Induction Service (KGIS), an end-to-end knowledge induction system, which provides numerous functionalities, most notably automatic taxonomy induction. The *learned* taxonomy is instantiated via a `Smart Spreadsheet`, which allows *users* to make changes as and how they see fit. The KGIS framework reduces the need of costly human annotations (i.e. it can automatically induce

Corpus: [NeurIPS papers \(1987 to 2017\)](#)

toggle comments
toggle control panel

	A	B	C	D	E	F	G
1	algorithm (Q8366)						
2	hybrid algorithm						
3	stochastic algorithm						
4	DP algorithm						
5	sorting						
6	sparse autoencoders						
7	spectral						
8	spectral gap						
9	spectral norm						
10	spectral method						
11	spectral radius						
12	spectral domain						
13	spectral methods						
14	spectral density						
15	spectral hashing						
16	Spectral methods						
17	spectral learning						
18							
19							
20							

(a) AutoComplete feature in Smart Spreadsheet.

Corpus: [NeurIPS papers \(1987 to 2017\)](#)

toggle comments
toggle control panel

	A	B	C	D	E	F	G
1	algorithm (Q8366)						
2	hybrid algorithm						
3	stochastic algorithm						
4	DP algorithm						
5	sorting						
6	sparse autoencoders						
7	spectral gap						
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							

Insert row above
 Insert row below
 Remove rows
 Populate from Seeds
 Populate from KG type
 Populate from KG, Distant Supervision
 Export to WKS

(b) Smart Spreadsheet Feature: Allowing automatic population.

Figure 4: Features for Smart Spreadsheet.

taxonomies), but at the same time allows for a *human-in-the-loop* to interact with it, thereby ensuring that the user has always the final say in all the system outputs.

In the future, we plan on developing customer-centric downstream applications for using this framework. In addition, we also plan on working upon additional knowledge centric problems, such as *Unsupervised Relation Induction* to provide additional facets to our proposed framework.

References

- Md. Faisal Mahbub Chowdhury and Robert Farrell. 2019. [An efficient approach for super and nested term indexing and retrieval](#). *CoRR*, abs/1905.09761.
- Marti A. Hearst. 1992. [Automatic acquisition of hyponyms from large text corpora](#). In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. 1990. [Introduction to WordNet: an on-line lexical database](#). *International Journal of Lexicography*, 3(4):235–244.

- Denny Vrandečić. 2012. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st international conference on world wide web*, pages 1063–1064. ACM.