AutoKnow: Self-Driving Knowledge Collection for Products of Thousands of Types

Dong et al.

Presented by Luke Song

Abstract

- Knowledge graph for products offered at online retail store (Amazon)
- Presents AUTOKNOW, automatic system that can conduct
    A. taxonomy construction,
    B. product property identification
    C. knowledge extraction
    D. anomaly detection
    E. synonym discovery
- System is
    A. Automatic: Requires little human intervention
    B. Multi-scalable: Scalable in multiple dimensions (domains, products, and attributes)
    C. Integrative: Exploit rich customer behavior logs

Outline

1. Background
2. Key Ideas
3. Model Architecture
4. Experimental Results
5. Conclusion
6. Discussion Time :)

Background

- Knowledge graph: Describes entities and relations between them
- Challenges for KG in the retail domain
    A. Structure-sparsity: Thousands of product attributes, billions of existing products, and millions of new products emerging on a daily basis without consistent structure
    B. Domain-complexity: Number of product types is towards millions and there are various relationships between the types like sub-types, product attributes vastly differ between types
    C. Product-type-variety: Product attributes, value vocabularies, text patterns in product titles and descriptions often differ for different types

Key Ideas

Background - Few key techniques

- Leverage graph structure that naturally applies to knowledge graphs and taxonomy, and apply Graph Neural Network
- Take product categorization as input signals to train models, and combine our tasks with product categorization for multi-task training to allow better performance
- Learn with limited labels to alleviate the burden of manual training data creation, relying heavily on weak supervision (e.g., distant supervision) and on semi-supervised learning
- Mine both facts and heterogeneous expressions for the same concept (i.e., type, attribute value) from customer behavior logs, abundant in the retail domain

Key Ideas

Contribution

- Operational system: Describes AutoKnow, a comprehensive end-to-end solution for product knowledge collection, covering components from ontology construction and enrichment, to data extraction, cleaning, and normalization
- Technical novelty: Range from NLP and graph mining to anomaly detection, and leverage state- of-the-art techniques in GNN, transformer, and multi-task learning
- Empirical Study: Able to extend the existing ontology by 2.9X, and considerably increase the quality of structured data, on average improving precision by 7.6% and recall by 16.4%

Key Ideas

Terminology

- Knowledge graph: set of triples in the form of (subject, predicate, object)
- Focuses on specific type of KG called *broad graph*.
  - Bipartite graph G = (N1, N2, E), where nodes in N1 represent entities of one particular type, called the topic type, nodes in N2 represent attribute values (that can be entities or atomic values), and edges in E connect each entity with its attribute values
- Two sources of input: **product catalog** that includes product taxonomy, a set of product attributes, a set of products, and attribute values for each product and **customer behavior logs**, such as the query and purchase log, customer reviews, and Q&A

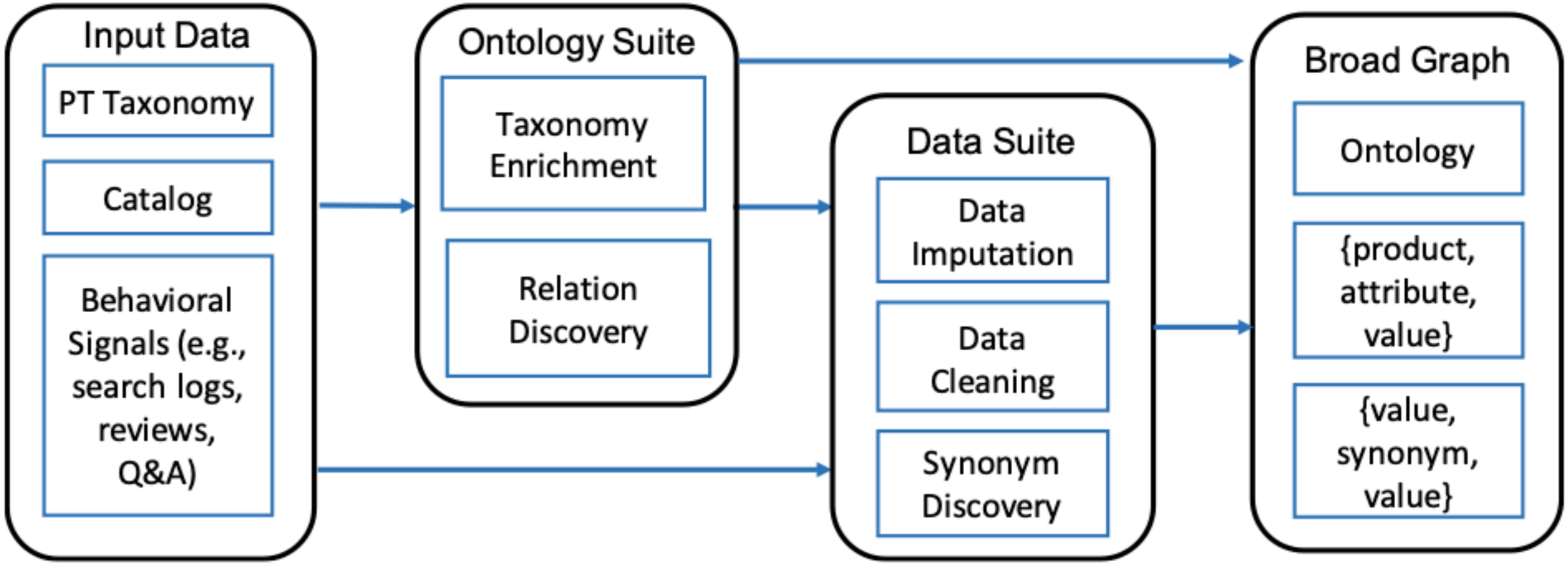
System Architecture

Formal Problem Definition

- Let C = (T, A, P) be a product catalog,
    1. *T* = (T, H) denotes a product taxonomy with a set of product types T and the hypernym relationships H between types in T
    2. *A* denotes a set of product attributes
    3. *P* = {PID, {T}, {(A,V)}} contains for each product (PID is the ID) a set of product types {T } and a set of attribute-value pairs {(A, V)}.
- Let L denote customer behavior logs
- *Product Knowledge Discovery* takes C and L as input, enriches the product knowledge by adding new types and hypernym relationships to T, and new product types and attribute values for each product in P.

System Architecture

High Level View



System Architecture

High Level View

- Ontology suite: **Taxonomy enrichment** identifies new product types not existing in input taxonomy T and decides the hypernym relationships between the newly discovered types and existing types, using them to enrich T. **Relation discovery** decides for each product type $T \in \mathcal{T}$ and attribute $A \in \mathcal{A}$, whether A applies to type T and if so, how important A is when customers make purchase decisions for these products, captured by an importance score
- Data suite: The data suite contains three components: **Data imputation**, **Data cleaning**, and **Synonym discovery**. Data imputation derives new (attribute, value) pairs for each product in P from product profiles and existing structured attributes. Data cleaning identifies anomalies from existing data in P and newly imputed values. Synonym discovery associates synonyms between product types and attribute values.

System Architecture

Ontology Suite - Taxonomy Enrichment

- Given an existing product taxonomy $\mathcal{T} = (T, H)$, extend it with $\mathcal{T} = (T \cup T', H \cup H')$, where T' is the set of new product types, and H' is the additional set of hypernym relationships between types in T and in T'
- First discover new types T' from product titles or customer search queries by training a **type extractor**. Then, **attach candidate types** in T' to the existing taxonomy T by solving a hypernym classification problem

System Architecture

Ontology Suite - Taxonomy Enrichment - Type Extractor

- Type extractor: Formulate type extraction as a “BIOE” sequential labeling problem
- To train, adopt distant supervision to generate the training labels
- For product titles, look for product types in Catalog provided by retailers (restricted to the existing product types), and generate BIOE tags when types are explicitly and exactly mentioned in their titles. For queries, look for the type of purchased products in the query to generate BIOE tags
- New types from titles are taken as T, and those from queries, albeit noisier, will be used for type attachment

System Architecture

Ontology Suite - Taxonomy Enrichment - Type Attachment

- Type attachment: solve a binary classification problem, where the classifier determines if the hypernym relationship exists between two types $T \in \mathrm{T}$, $T' \in \mathrm{T}'$

1. Construct a graph where the nodes represent types, products, and queries, and the edges represent various relationships including 1) product co-viewing, 2) a query leading to a product purchase, 3) the type mentioned in a query or a product (according to the extraction)
2. The type representation for each $T \in \mathrm{T} \cup \mathrm{T}'$ is combined with semantic features (e.g., word embedding) of the type names and fed to the classifier

- Use distant supervision to train: use the type hypernym pairs in the existing taxonomy as the supervision to generate positive labels, and generate five negative labels by randomly replacing the hyponym type with other product types

System Architecture

Ontology Suite - Relation Discovery

- Given a product taxonomy $T = (T, H)$ and a set of product attributes A, decides for each $(T , A) \in T \times A$, (1) whether attribute A applies to products of type T, denoted by $(T , A) \rightarrow \{0, 1\}$, and (2) how important A is for purchase decisions
- Train a classification model (random forest) using two types of features:
    1. Seller behavior: Captured by coverage of attribute values for a particular product type, and frequency of mentioning attribute values in product profiles
    2. Buyer behavior: Captured by frequency of mentioning attribute values in search queries, reviews, Q&A sessions, etc.
- Train a classification model to decide attribute applicability, and a regression model to decide attribute importance

System Architecture

Data Suite - Data Imputation

- Given product information (PID, {T }, {(A, V)}), extracts new (A, V) pairs for each product from its profiles (i.e., title, description, and bullets)

$$(y\_1,y\_2,...y\_L) = CRF(CondSelfAtt(BiLSTM(ex\_1,ex\_2,...,e\_xL),e\_T))$$

- Where e_T is the pre-trained hyperbolic-space embedding of product type T , known to preserve the hierarchical relation between taxonomy nodes. CondSelfAtt is the conditional self attention layer that allows e_T to influence the attention weights.
- Trains sequence tagging and product categorization at the same time, with a shared BiLSTM layer to better identify tokens that indicate the product type, and address the problem that products can be mis-classified or product type information can be missing in a catalog
- Adopt a distant supervision approach to automatically generate the training sequence labels by text matching between product profiles and available attribute values in the catalog

System Architecture

Data Suite - Data Cleaning

- Given product information (PID, {T }, {(A, V)}), identifies (A, V) pairs that are incorrect for the product, such as (A = flavor, V = “1 lb. box”) for a box of chocolate and (A = color, V = “100% Cotton”) for a shirt
- Propose transformer-based neural net model that jointly processes signals from textual product profile (D) and the product taxonomy T via a multi-head attention mechanism
- Input is the concatenation of token sequences in D, T andV
- Embedding e_i is generated from the addition of pre-trained FastText embedding, segment embedding vector that indicates to which source sequence (D, T or V) token i belongs, and positional embedding indicates the relative location of token i in the sequence
- Passes through a dense layer followed by a sigmoid node to produce a single score between 0 and 1
- Distant supervision to automatically generate training labels from the input Catalog. We generate positive examples by selecting high-frequency values that appear in multiple brands, then for each positive example we randomly apply one of the following three procedures to generate a negative example: 1) We build a vocabulary vocab(A) for each attribute A and replace a catalog value V of A with a randomly selected value from vocab(A); 2) We randomly select n-grams from the product title that does not contain the catalog value V , where n is a random number drawn according to the distribution of lengths of tokens in vocab(A); 3) We randomly pick the value of another attribute A′ A to replace V

System Architecture

Data Suite - Synonym Finding

1. Apply collaborative filtering on customer co-view behavior signals to retain product pairs with high similarity, and take their attribute values as candidate pairs for synonyms.
2. Train a simple logistic regression model to decide if a candidate pair has exactly the same meaning

- Features used include edit distance, pre-trained MT-DNN model score, and features regarding distinct vs. common words
- Distinct vs. common words play a critical role, which focuses on three sets of words: words appearing only in the first candidate but not the second, and vice versa, and words shared by the two candidates. Between every two out of these three sets, edit distance and embedding similarity are computed and used as features

Experimental Results

- Size of raw data is huge:

Product Domain	Grocery	Health	Beauty	Baby
#types	3,169	1,850	990	697
med. # products/type	1,760	18,320	27,150	28,700
#attributes	1,243	1,824	1,657	1,511
med. #attrs/type	113	195	228	206

- Performance evaluated on:
    1. triples with product types such as (product-1, hasType, Television)
    2. triples with attribute values such as (product-2, hasBrand, Sony)
    3. triples depicting entity relations such as (chocolate, isSynonymOf, choc)
- New metrics:
    - Defect rate: percentage of (product, attribute) pairs with missing or incorrect values
    - Applicability: percentage of products where attribute A applies

Experimental Results

Product Type Triples

Product Domain	Grocery	Health	Beauty	Baby
#types	3,169	1,850	990	697
med. # products/type	1,760	18,320	27,150	28,700
#attributes	1,243	1,824	1,657	1,511
med. #attrs/type	113	195	228	206

Experimental Results

Attribute Triples

	Attribute 1		Attribute 2		Attribute 3	
Grocery	Input	PG	Input	PG	Input	PG
Applicability	38.51%		7.53%		10.00%	
Precision	68.61%	82.59%	49.94%	77.30%	55.10%	55.10%
Recall	37.17%	83.15%	1.43%	80.96%	54.58%	54.59%
F-measure	48.22%	82.87%	2.78%	79.09%	54.84%	54.85%
Defect Rate	62.91%	21.14%	98.58%	30.72%	49.50%	49.49%
Health	Input	PG	Input	PG	Input	PG
Applicability	1.35%		0.59%		57.92%	
Precision	70.54%	84.00%	59.11%	70.00%	78.00%	61.75%
Recall	59.69%	49.92%	69.50%	63.45%	47.13%	69.92%
F-measure	64.66%	62.62%	63.89%	66.56%	58.76%	65.58%
Defect Rate	49.69%	52.34%	48.31%	45.45%	55.04%	38.28%
Beauty	Input	PG	Input	PG	Input	PG
Applicability	0.04%		0.54%		4.82%	
Precision	18.83%	48.00%	71.98%	76.00%	62.00%	61.68%
Recall	69.44%	69.44%	65.21%	59.95%	53.26%	62.98%
F-measure	29.62%	56.76%	68.43%	67.03%	57.30%	62.32%
Defect Rate	82.35%	59.02%	40.19%	42.76%	48.51%	39.50%
Baby	Input	PG	Input	PG	Input	PG
Applicability	0.0011%		0.09%		55.82%	
Precision	1.45%	0.03%	8.22%	10.60%	42.00%	49.54%
Recall	9.79%	9.79%	3.83%	50.92%	44.13%	56.39%
F-measure	2.53%	0.06%	5.23%	17.55%	43.04%	52.74%
Defect Rate	98.72%	99.97%	97.30%	89.63%	60.81%	50.46%

Grocery	**Recall@ 90%**	**Recall@ 80%**	**#Removed**	**%Removed**
Attribute 1	36.58%	58.20%	1,381,277	55.06%
Attribute 2	9.92%	13.29%	320,960	59.40%
Health	**Recall@ 90%**	**Recall@ 80%**	**#Removed**	**%Removed**
Attribute 1	76.72%	85.93%	30,215	32.63%
Attribute 2	3.18%	21.14%	14,110	20.92%
Beauty	**Recall@ 90%**	**Recall@ 80%**	**#Removed**	**%Removed**
Attribute 1	94.33%	97.16%	10,926	62.31%
Attribute 2	46.05%	69.74%	7,651	11.87%
Baby	**Recall@ 90%**	**Recall@ 80%**	**#Removed**	**%Removed**
Attribute 1	87.24%	95.31%	2,673	66.17%
Attribute 2	52.07%	59.24%	1,956	73.04%

Experimental Results

Relation Triples

	Attribute 1	Attribute 2	Product Types
Precision	91.6%	93.7%	88.1%
#Pairs	6,610	1,066	21,900

Conclusion

- May need to fundamentally reconsider the way we model taxonomy and classify products b/c it’s not usually a tree structure
- Large volume of noises can deteriorate the performance of the imputation and cleaning models
- Need more data than just the text data