Abstractive Tabular Dataset Summarization via Knowledge Base Semantic Embeddings

Paul Azunre PAUL@NEWKNOWLEDGE.IO
Craig Corcoran CRAIG@NEWKNOWLEDGE.IO
David Sullivan DAVID@NEWKNOWLEDGE.IO
Garrett Honke GARRETT@NEWKNOWLEDGE.IO
Rebecca Ruppel REBECCA@NEWKNOWLEDGE.IO
Sandeep Verma SANDEEP@NEWKNOWLEDGE.IO
Jonathon Morgan JONATHON@NEWKNOWLEDGE.IO
New Knowledge, Austin, Texas, USA

Scott Langevin SLANGEVIN@UNCHARTED.SOFTWARE
Uncharted Software Inc., Toronto, ON, Canada

Editor: Editor's name

Abstract

This paper describes an abstractive summarization method[1] for tabular data which employs a knowledge base semantic embedding to generate the summary. Assuming the dataset contains descriptive text in headers, columns and/or some augmenting metadata, the system employs the embedding to recommend a subject/type for each text segment. Recommendations are aggregated into a small collection of *super types* considered to be descriptive of the dataset by exploiting the hierarchy of types in a prespecified ontology. We present experimental results on open data taken from several sources – OpenML, CKAN and data.world – to illustrate the effectiveness of the approach.

Keywords: Dataset Summarization, Type Recommendation, Semantic Embeddings

1. Introduction

The motivation of this work is to develop a method for summarizing the content of tabular datasets. One can imagine the potential utility of automatically assigning a set of tags to each member of a large collection of datasets that would indicate the potential subject being addressed by the dataset. This can allow for semantic querying over the dataset collection to extract all available data pertinent to some specific task subject at scale.

We make the assumption that the dataset contains some text that is semantically descriptive of the dataset subject, whether appearing in columns, headers or some augmenting metadata. As opposed to an *extractive* approach that would merely select some exact words and phrases from the available text, we propose an *abstractive* approach that builds an internal semantic representation and produces subject tags that may not be explicitly present in the text augmenting the dataset.

The result of this work is *DUKE*—**D**ataset **U**nderstanding via **K**nowledge-base **E**mbeddings—a method that employs a pretrained Knowledge Base (KB) semantic embedding to perform

1. Our code is available for download at https://github.com/NewKnowledge/duke

© P. Azunre, C. Corcoran, D. Sullivan, G. Honke, R. Ruppel, S. Verma, J. Morgan & S. Langevin.

type recommendation within a prespecified *ontology*. This is achieved by aggregating the recommended types into a small collection of *super types* predicted to be descriptive of the dataset by exploiting the hierarchical structure of the various types in the ontology. Effectively, the method represents employing an existing KB embedding to extensionally generate a *dataset2vec* embedding. Using a February 2015 Wikipedia knowledge base and a corresponding DBpedia ontology to specify types, we present experimental results on open data taken from several sources—OpenML, CKAN, and data.world—to illustrate the effectiveness of the approach.

2. Related Work

The *distributional semantics* (Sahlgren, 2008) concept has been recently widely employed as a natural language processing (NLP) tool to embed various NLP concepts into vector spaces. This rather intuitive hypothesis states that the meaning of a word is determined by its context. By far the most pervasive application of the hypothesis has been the word2vec model (Mikolov et al., 2013)(Pennington et al., 2014) which employs neural networks on large corpora to embed words that are contextually similar to be close to each other in a high-dimensional vector space. Arithmetic operations on the elements of the vector space produce semantically meaningful results, e.g., *King-Man+Woman=Queen*.

Since the original word2vec model, various incremental incarnations of it have been employed to embed sentences, paragraphs and even knowledge graphs into vector spaces via sent2vec (Pagliardini et al., 2017), paragraph2vec (Le and Mikolov, 2014), and RDF2Vec (Ristoski and Paulheim, 2016) respectively.

A topic *domain* is typically expressed as a manually curated ontology. A basic element of an ontology is a *type*, and a *type assertion* statement links specific entities of the knowledge graph to specific types. These statements can be used to augment a semantic embedding space with type information in order to add high level graph context to the embedding space. For instance, it was recently shown that one can extend a pretrained Knowledge Graph Embedding (KGE) to contain types of a specific ontology if those were not already present as entities, given a list of assertion statements (Kejriwal and Szekely, 2017). Thus, it can be assumed that a semantic embedding is *typed* for our purposes.

We note that the abstractive tabular dataset summarization problem is closely related to the well-studied problem of *type recommendation*, where the type is a super tag for all text segments in the dataset within a prespecified ontology that needs to be predicted. Systems for type recommendation using both manually curated features (Ma et al., 2013) and automated features (van Erp and Vossen, 2017), e.g., via typed KGEs (Kejriwal and Szekely, 2017), for *individual entities*, have been previously explored. To the best of our knowledge, this is the first application of typed semantic embeddings to abstractive tabular dataset summarization.

3. Approach

3.1. Framework

In this subsection, we present a pair of definitions to aid orientation.

Definition (word2vec) Word2vec models utilize a large corpus of documents to build a vector space mapping words to points in a space, where proximity implies semantic similarity (Mikolov et al., 2013).

Definition (wiki2vec) A *wiki2vec* model is a form of word2vec model trained on a corpus of Wikipedia KB text documents. Note that we used a pre-trained 1000 dimensional skip-gram model[2] with no stemming and window size 10 for all of our experiments. Also note that wiki2vec is different from a KGE, which is typically trained on relationship triples between entities in a knowledge graph (Ristoski and Paulheim, 2016).

3.2. Generating Type Recommendations

The method for summarizing a tabular dataset can be broken down into three distinct steps:

1. Collect a set of *types* and an *ontology* to use for abstraction
2. Extract any text data from the tabular dataset and embed it in a vector space, calculating the similarity between each text segment and each type in the ontology
3. Aggregate these similarity vectors into a single vector of similarities

3.2.1. Type Ontology

In order to generate an abstract term to describe the dataset, we must first collect an ontology of types to select a descriptive term from. We use an ontology provided by DBpedia[3] which contains $M \approx 400$ defined types, including everything from *sound* to *video game* and *historic place*. DBpedia also contains defined parent-child relationships for the types[4] that we use to build a complete hierarchy of types e.g. that *tree* is a sub-type of *plant* which is a sub-type of *eukaryote*.

3.2.2. Word Embedding

With the ontology defined, extract each word from the dataset, embed it in a wiki2vec vector space, and calculate the similarity between that word and every type in the ontology. The result is a vector $v \in \mathbf{R}^M$ for each word w, where each v_i is the similarity between w and topic i. If a single cell in a column contains more than one word, take the average of the corresponding embedded vectors. Gather vectors v for each column c into a matrix of similarities D^c, where d_{ij} represents the similarity between cell j and topic i. If column headers are provided, treat them as an additional column in the dataset.

3.2.3. Similarity Aggregation

The goal of this step is to reduce matrices D^c to a single vector of similarities.

We utilize three successive aggregations in order to compute this final vector. The first transforms D^c into a vector $u^c \in \mathbf{R}^M$ by operating over its rows. The second *tree* aggregation updates u^c based on the hierarchy of ontology types. For instance, the score for *means of transportation* may be updated based on the scores for *airplane*, *train*, and

2. Available at https://github.com/idio/wiki2vec
3. Downloaded from http://downloads.dbpedia.org/2015-10/dbpedia 2015-10.nt
4. Defined parent-child type relationships can be found at http://dbpedia.org/ontology/

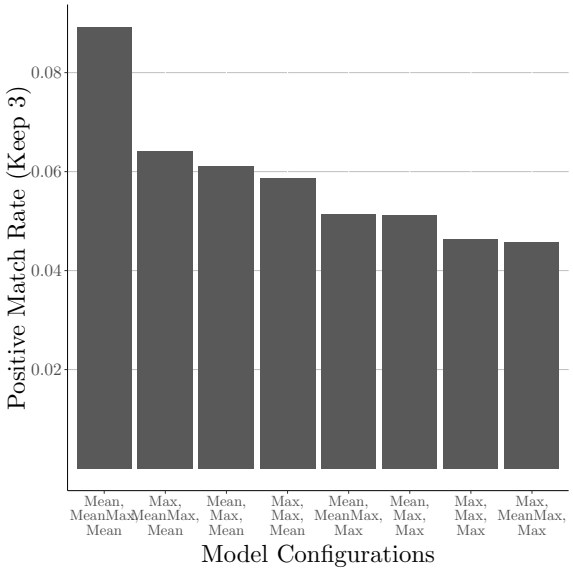

Figure 1: Match rate between true labels and top 3 predicted labels for the best performing aggregation function combinations. The labels for each bar describe the three tested aggregation functions in the order: column, tree, dataset.

automobile. After combining vectors u^c into a matrix E, the third aggregation operates over its rows to yield a single vector $t \in \mathbf{R}^M$ that represents similarities between the dataset as a whole and each type in the ontology.

We tested two simple functions for each aggregation step: *mean* and *max*, as well as a variety of more complex aggregations for the tree aggregation step. We found that the most successful tree aggregation functions utilize different sub-functions for processing child scores and the original score, for e.g., as shown in Equation (2) below.

$$h(i,j) := \begin{cases} u_j, & \text{if type } j \text{ is a child of type } i. \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

$$u_i = \frac{1}{2}\left(u_i + max_j\left(h\left(i,j\right)\right)\right) \tag{2}$$

Here, h is a utility function returning similarity u_j if j is a child of i.

3.2.4. Aggregation Function Selection

To select the best function for each aggregation step, we first hand-labeled a collection of datasets with types from the ontology. For each combination of aggregation functions, we then computed the percentage of true labels found in the top three labels predicted by DUKE (results shown in Figure 1). We found that using *mean* for column aggregation, *mean-max* tree aggregation update described by Equation (2), and *mean* for the final dataset aggregation step produces the best results.

Table 1: CKAN tabular dataset summarization results

Dataset	Curated Tags	Ontology Tags	DUKE Tags
Class Size 2016-2017	class size, public, school, students in classes	educational institution, school	educational institution, school, public service
2016 Annual Survey Questions	annual survey, library, public library, public library	library	library
BC Liquor Store Product Price List Oct 2017	BC Liquor Stores, alcohol, beer, price, beverage, wine, spirits	wine, beer, vodka, beverage, wine region, brewery, winery	wine, beer, vodka, beverage, wine region, winery, grape, controlled designation of origin wine
Coalfile Report	assessment reports, coal, data, maps	mine, coal pit	river

4. Results and Discussion

The goal of this section is to illustrate the effectiveness of the proposed approach in the context of some open data sets for which manually curated tags are available to facilitate evaluation. Names of datasets are provided to facilitate verification. For each dataset, we manually translate curated tags into tags within our ontology. We then generate tags using DUKE, and grade it using standard Precision and Recall metrics. When generating tags with DUKE, we return all tags with dataset similarities greater than 0.25 (this number is based on experimental testing experience). If this yields an empty set, we only return the tag with the highest dataset similarity. Moreover, we plot the top 5 DUKE-predicted tags and the original curated tags for three datasets in Figure 2, for visual investigation. Each dataset takes approximately 3 seconds to analyze serially on a 16 CPU 64 GB D16s v3 Azure Cloud VM.

4.1. Example 1 - CKAN Datasets

Four randomly selected CKAN datasets were used: Class Size 2016-2017, 2016 Annual Survey Questions, BC Liquor Store Product Price List Oct 2017, and Coalfile Reports. Experimental results of running DUKE on these are shown in Table 1. Precision of 0.83 and recall of 0.77 are achieved.

Analyzing these results, we see only one egregiously erroneous DUKE tag: *river* for *mine/coal pit*. Investigating further into the data, we found *river* to be a common semantic theme in coal field names (and presumably location, examples include *Elk River*, *Hat Creek* and *Peace River*), validating the observed result. However, the low dataset similarity value of 0.19 for this prediction would have alerted the user to the low quality of this DUKE tag.

4.2. Example 2 - OpenML Datasets

Four simple OpenML datasets were obtained through the D3M DARPA program: the 185 baseball, 196 autoMpg, 30 personae, and 313 spectrometer datasets. The results for these datasets are shown in Table 2. Precision and Recall were both calculated to be 0.67.

Table 2: OpenML tabular dataset summarization results

Dataset	Curated Tags	Ontology Tags	DUKE Tags
185 baseball	baseball player, play statistics, database	baseball player	baseball player
196 autoMpg	city-cycle, miles per gallon, fuel consumption	engine, automobile engine automobile	engine, fuel type, automobile engine, engine configuration
30 personae	personality, prediction, from text	person	person
313 spectrometer	measurement, sky, red band, blue band, spectrum, database, flux	colour	band

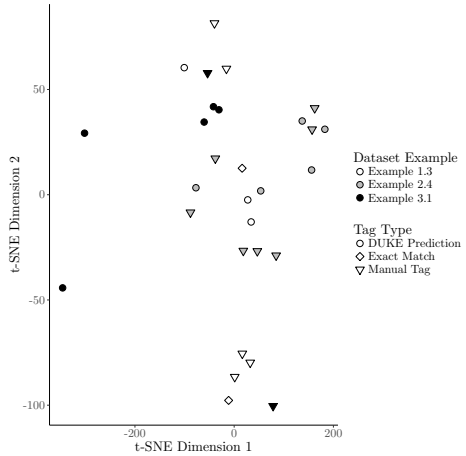

Figure 2: Concept embedding space for three of the examined datasets. Point shape depicts DUKE predictions and manual tags. t-SNE dimension reduction was used to project the 1000 dimension concept embeddings into a 2D space for presentation.

4.3. Example 3 - data.world Datasets

Names of 4 randomly-selected data.world datasets are: US terrorist origins, Occupational Employment Growth, CAFOD Activity File Haiti and Queensland Gambling Data. Results are not shown in detail, due to space constraints. Precision was 0.71 and Recall was 0.83.

5. Conclusion

Results of numerical experiments show good agreement between manual and tags generated via the abstractive summarization method presented. Results can be improved by including more refined ontologies, retraining wiki2vec on more complete versions of DBpedia (potentially augmented as in (Wang et al., 2015) (Groth et al., 2016)), and more sophisticated multi-word phrase handling.

Acknowledgments

Work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract Number D3M (FA8750-17-C-0094). Views, opinions, and findings contained in this report are those of the authors and should not be construed as an official Department of Defense position, policy, or decision.

References

Paul T. Groth, Sujit Pal, Darin McBeath, Brad Allen, and Ron Daniel. Applying universal schemas for domain specific ontology expansion. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016, San Diego, CA, USA, June 17, 2016*, pages 81–85, 2016. URL http://aclweb.org/anthology/W/W16/W16-1315.pdf.

Mayank Kejriwal and Pedro Szekely. Supervised typing of big graphs using semantic embeddings. In *Proceedings of The International Workshop on Semantic Big Data*, SBD '17, pages 3:1–3:6, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4987-1. doi: 10.1145/3066911.3066918. URL http://doi.acm.org/10.1145/3066911.3066918.

Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014. URL http://arxiv.org/abs/1405.4053.

Y. Ma, T. Tran, and V. Bicer. Typifier: Inferring the type semantics of structured data. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 206–217, April 2013. doi: 10.1109/ICDE.2013.6544826.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *CoRR*, abs/1703.02507, 2017. URL http://arxiv.org/abs/1703.02507.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

Petar Ristoski and Heiko Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In Paul Groth, Elena Simperl, Alasdair Gray, Marta Sabou, Markus Krötzsch, Freddy Lecue, Fabian Flöck, and Yolanda Gil, editors, *The Semantic Web – ISWC 2016*, pages 498–514, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46523-4.

Magnus Sahlgren. The distributional hypothesis. *Italian Journal of Linguistics*, 20(1): 33–54, 2008.

Marieke van Erp and Piek Vossen. Entity typing using distributional semantics and dbpedia. In Marieke van Erp, Sebastian Hellmann, John P. McCrae, Christian Chiarcos, Key-Sun Choi, Jorge Gracia, Yoshihiko Hayashi, Seiji Koide, Pablo Mendes, Heiko Paulheim, and Hideaki Takeda, editors, *Knowledge Graphs and Language Technology*, pages 102–118, Cham, 2017. Springer International Publishing. ISBN 978-3-319-68723-0.

Quan Wang, Bin Wang, and Li Guo. Knowledge base completion using embeddings and rules. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1859–1866, 2015. URL http://ijcai.org/Abstract/15/264.