AYNEC-DataGen: a tool for generating evaluation datasets for Knowledge Graphs completion

Daniel Ayala[1(✉)], Agustín Borrego[1], Inma Hernández[1], Carlos R. Rivero[2], and David Ruiz[1]

[1] University of Seville, Seville, Spain
{dayala1, borrego, inmahernandez, druiz}@us.es
[2] Rochester Institute of Technology, Rochester, NY, USA
crr@cs.rit.edu

Abstract. In the context of knowledge graphs, the task of completion of relations consists in adding missing triples to a knowledge graph, usually by classifying potential candidates as true of false. Creating an evaluation dataset for these techniques is not trivial, since there is a large amount of variables to consider which, if not taken into account, may cause misleading results. So far, there is not a well defined workflow that identifies the variation points when creating a dataset, and what are the possible strategies that can be followed in each step. Furthermore, there are no tools that help create such datasets in an easy way. To address this need, we have created AYNEC-DataGen, a customisable tool for the generation of datasets with multiple variation points related to the preprocessing of the original knowledge graph, the splitting of triples into training and testing sets, and the generation of negative examples. The output of our tool includes the evaluation dataset, an optional export in an open format for its visualisation, and additional files with metadata. Our tool is freely available online.

Keywords: Knowledge Graph · Graph refinement · Evaluation · Tool

1 Introduction

Knowledge graphs are representations of knowledge as a collection of entities (nodes) and relations that connect them (edges). Each connection between entities represents a fact in the form of a <source, relation, target> triple. Knowledge graphs have enjoyed an increase of popularity in the recent years thanks to the development and use of knowledge graphs to store and connect structured information [2, 8]. Knowledge graphs are usually created through unsupervised means and are prone to errors [6], which is why there is also an increased interest in refinement techniques that either remove false facts, or infer missing ones.

Inferring missing facts from a knowledge graph is known as graph completion, which can be seen as the classification of potential triples as true or false.

The evaluation of graph completion techniques is usually performed by splitting a known knowledge graph into a training and testing set. The training set is used to learn a model that is able to classify the triples in the testing set. The testing set represents the missing knowledge from the training set. Apart from the splitting, the knowledge graph used for evaluation must undergo a preprocessing and negatives generation steps. The former transforms the graph in any way that is considered convenient. The latter generates negative examples that are crucial to learn and test the model.

In these three steps (preprocessing, splitting, and negatives generation), there is a large number of variation points where we can apply different strategies that vary according to the purposes of the experimentation and the knowledge graph being used. This makes dataset generation a difficult process where many considerations must be taken into account to avoid cases in which the results of the evaluation are unfairly good or bad. To streamline this process, we have created AYNEC-DataGen, a tool for the creation of evaluation datasets for knowledge graph completion that offers several configurable variation points, and is freely available online[1]. AYNEC is part of an evaluation suite that covers the entire evaluation workflow, from dataset creation to comparison of results [1].

This paper is organized as follows. Section 2 describes some of the approaches for evaluation followed by graph completion proposals. Section 3 describes the functionality of the tool. Section 4 summarises the conclusions.

2 Related work

To the best of our knowledge, there is no tool for the streamlined generation of evaluation datasets for knowledge graph completion. Each proposal describes their evaluation setup in a different way. Some popular variation points are the fraction of the knowledge graph used for testing, the number of negative examples to generate for each positive triple, and the strategy used to generate negatives. For example, Socher et al. [7] use 10% of the graph for testing, and generate one negative per positive by replacing the target of positive triples. Gardner and Mitchell [3] use 25% of the graph for testing, and generate a variable number of negatives (usually more than 5) by replacing both the source and target of positive triples. Ji et al. [4] use 20% of the graph for testing, generating the negatives in the same way as Socher et al. [7]. Finally, Mazumder and Liu [5] use 20% of the graph for testing, and generate 4 negatives per positive, 2 by replacing the source, and 2 by replacing the target.

The lack of proper tools that focus on this aspect leads to overlooking some aspects. For example, all the aforementioned proposals do not deal with the presence of inverse relations in a dataset, such as "parent" and "child". These make completion a trivial task in which edges are infered by checking whether or not the inverse edge exists. The used datasets, such as Freebase [3–5, 7], NELL [3] or WordNet [4, 5, 7], contain a high proportion of inverse relations. The lack of

[1] https://github.com/tdg-seville/AYNEC

this and other considerations when it comes to creating datasets motivated us to create AYNEC-DataGen.

3 Our tool

Our tool, AYNEC-DataGen, takes as input a knowledge graph and the configuration of its variation points, and outputs the evaluation dataset along with several metadata and summary files. It is implemented as a Python script with command-line arguments that allows its external use. For example, we have also implemented a Web interface with a form used to set argument values.

3.1 Input

The input knowledge graph can be given in several popular formats, such as n-triples, RDF, or text files with a triple per line with source, relation and target separated by spaces. The configurable options can be separated into preprocessing, split, and negatives generation. They are implemented in a modular way, so that it is possible to add new options or strategies to any variation point.

Regarding preprocessing, the variation points are the following: the overall fraction of the graph to keep (useful for reducing the size of the graph), the minimum frequency a relation must reach to be kept in the dataset, the accumulated fraction of the edges that must be kept in the dataset, the overlap threshold to be used to detect inverses, whether or not inverse relations should be removed, and whether or not type-denoting relations should be set apart when generating the evaluation dataset.

Regarding splitting, the only variation point is the percentage of the graph that will be used for testing. AYNEC-DataGen takes the same percentage from each relation, though in-depth configuration allows the use of different percentages for each relation.

Regarding negatives generation, the variation points are the following: whether or not negatives should be generated in the training set (some proposals generate their own negatives), the number of negatives to generate per positive, and the strategy used to generate them. The supported strategies so far are the replacement of the source and/or target of the triples by keeping or bot the domain/range of the relation of the triple, and the PPR-based approach described by Gardner and Mitchell [3].

3.2 Output

The main output are the training and testing files, each of them containing triples labelled as positive (1) or negative (-1). In addition to these files, we include the following ones: two files listing each relation and its frequency, and each entity and its degree (inwards, outwards, and total); a HTML summary of the aforementioned frequencies; a file with the data properties of each entity in the dataset; a file with the inverse relations in the graph (even if they were

removed); a file with the types of each entity, if type relations were configured as set apart; and both the training and testing sets in a single .gexf file with labelled edges that enables the use of the dataset by external tools.

4 Conclusions

In this paper, we have presented AYNEC-DataGen, a tool for the generation of evaluation datasets for knowledge graph completion. Knowledge graph completion is one of the tasks that has recently surged in popularity thanks to the advent of knowledge graphs. However, being a relatively young field of research, evaluation is heterogeneous, and there is not a well defined workflow nor set of good practices when it comes to generating evaluation datasets.

Our tool covers this need by means of a highly configurable workflow that allows the customisation of the preprocessing, splitting, and negative generation steps. Our tool outputs, along with the evaluation datasets, a series of helpful metadata files, and a .gexf file for easy import into other graph-processing tools.

AYNEC-DataGen is publicly available online. We intend to maintain and expand it if new requirements are identified, such as novel negatives generation or graph splitting strategies.

Acknowledgements Our work was supported the Spanish R&D&I programme by grant TIN2016-75394-R.

References

1. D. Ayala, A. Borrego, I. Hernández, C. R. Rivero, and D. Ruiz. AYNEC: All you need for evaluating completion techniques in knowledge graphs. In *ESWC*, page to appear, 2019.
2. K. D. Bollacker, R. P. Cook, and P. Tufts. Freebase: A shared database of structured general human knowledge. In *AAAI 22*, pages 1962–1963, 2007.
3. M. Gardner and T. M. Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In *EMNLP*, pages 1488–1498, 2015.
4. G. Ji, S. He, L. Xu, K. Liu, and J. Zhao. Knowledge graph embedding via dynamic mapping matrix. In *ACL*, pages 687–696, 2015. https://doi.org/10.3115/v1/P15-1067.
5. S. Mazumder and B. Liu. Context-aware path ranking for knowledge base completion. In *IJCAIJ*, pages 1195–1201, 2017. https://doi.org/10.24963/ijcai.2017/166.
6. H. Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.
7. R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.
8. R. Speer and C. Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012.