AYNEC: All You Need for Evaluating Completion Techniques in Knowledge Graphs

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. The popularity of knowledge graphs has led to the development of techniques to refine them and increase their quality. One of the main refinement tasks is completion (also known as link prediction for knowledge graphs), which seeks to add missing triples to the graph, usually by classifying potential ones as true or false. While there is a wide variety of graph completion techniques, there is no standard evaluation setup, so each proposal is evaluated using different datasets and metrics. In this paper we present AYNEC, a suite for the evaluation of knowledge graph completion techniques that covers the entire evaluation workflow. It includes a customisable tool for the generation of datasets with multiple variation points related to the preprocessing of graphs, the splitting into training and testing examples, and the generation of negative examples. AYNEC also provides a visual summary of the graph and the optional exportation of the datasets in an open format for their visualisation. We use AYNEC to generate a library of datasets ready to use for evaluation purposes based on several popular knowledge graphs. Finally, it includes a tool that computes relevant metrics and uses significance tests to compare each pair of techniques. These open source tools, along with the datasets, are freely available to the research community and will be maintained.

Keywords: Knowledge Graph · Graph refinement · Evaluation · Datasets

1 Introduction

The recent years have seen an increase in popularity of the representation of large databases as graphs with nodes that represent entities and edges that represent relations between them. The advent of Linked Open Data [4], and the development of connected sources of structured data [1, 5, 13–15, 19, 22, 23] have drawn attention towards the use of knowledge graphs to represent knowledge, as well as the development of techniques that work on graph data [2, 10].

These graphs are not perfect and may be incomplete or contain errors [16]. The techniques that attempt to improve the quality of knowledge graphs are known in general as graph refinement proposals [16], a category that includes two types of proposals: those that detect incorrect information on graphs, and those that complete the graphs with missing information. We focus on the latter, also known as knowledge graph completion proposals, or link prediction for knowledge graphs. Adding missing knowledge to a graph might be seen as a binary classification problem in which the input is a triple $<s,r,t>$ (source entity, relation, target entity, also known as subject, predicate, object) that represents an edge in the graph, and the output is a binary value denoting whether or not that triple should be included in the graph.

There is a wide variety of graph completion proposals based on different approaches like embeddings [9, 21] or path-based features [8, 11]. Deep learning techniques have seen popularity in the recent years [18]. However, their evaluation is not homogeneous, that is, each proposal is evaluated on different knowledge graphs, using different methodologies and metrics to analyse results. There are well-known knowledge graphs that are commonly used for evaluation purposes, such as Freebase [8, 9, 11, 21] or WordNet [9, 11, 21]. However, when used for the specific task of evaluating graph completion proposals, these graphs are usually pre-processed by the different authors, who apply different criteria to obtain smaller and cleaner versions of the datasets, such as FB13 [21] or WN18 [6]. This makes it difficult to compare different proposals side by side, especially considering that evaluating graph refinement proposals is not trivial with many considerations and variants [16].

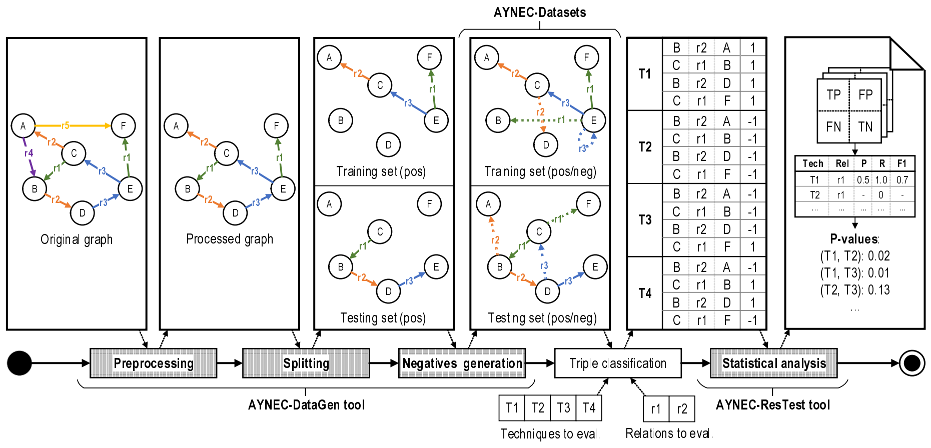
There is a clear need for a standard suite that defines both datasets and metrics to be used in the evaluation of graph completion proposals. To fulfill that need, in this paper we present AYNEC (**A**ll **Y**ou **N**eed for **E**valuating **C**ompletion), a resource for the evaluation of knowledge graph completion proposals that covers the entire evaluation workflow: preprocessing, training/testing splitting, generation of negative examples (which we refer to as negatives generation for the sake of brevity), and statistical analysis. The main contributions of AYNEC are: AYNEC-DataGen, a tool for the generation of evaluation datasets, which includes options for exporting the datasets in open formats for their easy visualisation, and offers several variation points in the preprocessing, splitting, and negatives generation steps; AYNEC-ResTest, a tool for computing metrics and significance tests from the results of several techniques in the statistical analysis step; and an initial collection of evaluation datasets generated from high quality subgraphs of popular knowledge graphs: WN11, WN18, FB13, FB15K, and NELL. If the specific datasets that we offer are not suited for a certain task, or if new requirements arise in the future, AYNEC-DataGen allows to easily extend the collection with new datasets.

Our datasets can be freely downloaded from Zenodo[1], under the CC BY 4.0 license. Our tools are open source and available as a public repository in GitHub[2] under the GPLv3 license. The source code of the tools is documented, describing each configurable parameter and function.

[1] http://doi.org/10.5281/zenodo.1744988
[2] https://github.com/tdg-seville/AYNEC

The rest of this paper is structured as follows: Section 2 describes each step in the evaluation workflow that we have identified, Section 3 shows what evaluation setups were used in several completion proposals in terms of the aforementioned workflow, Section 4 describes how AYNEC-DataGen implements the dataset creation steps, Section 5 describes AYNEC-ResTest, our metrics evaluation tool, Section 6 describes the specific datasets we propose for homogeneous evaluation, and Section 7 summarises our work and concludes the paper.

2 Workflow



T1	B	r2	A	1
	C	r1	B	1
	B	r2	D	1
	C	r1	F	1
T2	B	r2	A	-1
	C	r1	B	-1
	B	r2	D	-1
	C	r1	F	-1
T3	B	r2	A	-1
	C	r1	B	-1
	B	r2	D	-1
	C	r1	F	1
T4	B	r2	A	-1
	C	r1	B	1
	B	r2	D	1
	C	r1	F	-1

Tech	Rel	P	R	F1
T1	r1	0.5	1.0	0.7
T2	r1	-	0	-
...	...	...	...	...

Fig. 1. Knowledge graph completion evaluation workflow.

We have identified the necessary steps to evaluate graph completion proposals, and we have defined an abstract workflow that can be used to describe or compare different evaluation setups in the same terms.

Figure 1 depicts the evaluation workflow we have identified, which is composed of five steps. The external inputs are the original knowledge graph to use for evaluation, the techniques that are evaluated, and the relations on which the techniques will be evaluated. The final output is the comparison of the techniques according to a number of metrics and significance tests. For example, Gardner and Mitchell [8] take Freebase and NELL as original graphs, and evaluate two techniques on a total of 34 relations according to the metrics MAP and MRR.

The shadowed steps in Figure 1 (preprocessing, splitting, negatives generation, and metrics analysis) are the ones covered by our suite, while triple classification is the main task to be performed by each completion proposal.

Next, we discuss every step, its inputs and outputs, and we provide examples based on Figure 1.

2.1 Preprocessing

The goal of this step is to load and preprocess the original knowledge graph (which may have undergone previous preprocessing) in any way that is considered convenient. This can include, for example, the removal of relations with frequency below a threshold, the transformation of entity or relation names, or the insertion of new relations to enrich the graph.

Regarding the input, in some knowledge graphs, entities have literals attached that represent simple values related to the entity (e.g. data properties in DBpedia [1]). In other graphs, this kind of data is represented with additional nodes (e.g., a node that represents the year 2008, as happens in NELL [14]).

The **input** to this step is a knowledge graph that represents entities as nodes, and relations between them as edges. The **output** is an updated version of the original knowledge graph. In the **example** of Figure 1, relations with less than 20% of the total edges are removed.

2.2 Splitting

Evaluating techniques require using at least two sets: training and testing. The training set is the only part of the graph available when training a model, which is afterwards evaluated on the testing set.

The goal of the splitting step is to divide the edges in the original graph into two disjoint sets (training and testing). This way, we simulate a controlled scenario of incompleteness in which we know the missing edges (the testing set).

The testing set is usually a small fraction of the graph edges, ranging from 10% to 30%. Different strategies can be used to split the original dataset. For example, the edges taken for the testing set could be completely random, or we could take a fixed fraction of each relation for testing.

The **input** to this step is the preprocessed knowledge graph. The **outputs** are two disjoint sets of edges from the input graph, corresponding to the training and testing sets. In the **example** of Figure 1, 50% of the edges of each relation are taken for the testing set.

2.3 Negatives generation

The goal of this step is to generate negative triples, usually by creating new ones that are not found in the original knowledge graph.

The negative examples in the training set help learn a model. Their inclusion is optional, since the techniques themselves may be able to generate them.

The negative examples in the testing set are used to compute metrics like precision or recall. If negatives are not explicitly included in the testing set, any triple that is not found in the training or testing sets would be a negative when computing metrics. This would be the case, for example, if we want to test a technique that, instead of giving a score to input triples, outputs a set of triples as the positives, and assumes every other triple is negative.

The negative examples, when generated, should pose a challenge for the classification model [21]. For instance, it is trivial to classify <John-S., cousin-of, Republic-of-Guatemala> as false by checking that "Republic-of-Guatemala" is outside the range of relation "cousin-of". A more compelling example would be <John-S, cousin-of, Mary-S.>, in which the former sanity check is not enough.

The first point of variability when generating negative examples is how many of them should be generated. Usually, a fixed numbers of negative examples are generated per each positive example.

The second point of variability is the generation strategy. The idea is to take a positive example and change its source, target, or both, by choosing a replacement among a set of candidates. Some proposals [9, 11, 21] select as candidates the entities that are known to appear in the same position of the same relation, keeping the domain (if it is the source) or range (if it is the target) to avoid trivial cases like the <John-S., cousin-of, Republic-of-Guatemala> example. Others follow more elaborate approaches, such as giving a higher probability to nodes that are close to the original ones [8].

One of the problems when generating negative examples is that knowledge graphs are usually ruled by the open world assumption, which means that the absence of a triple in the graph does not necessarily mean that the triple is not true, just unknown. However, since knowledge graphs tend to be sparse, the probability of a true missing edge being chosen as a negative example should be negligible, which is why related datasets in practice follow a closed world assumption [8, 9, 11, 21].

The **inputs** to this step are the testing set and, optionally, the training set. The **outputs** are the sets with added negative edges. In the **example** of Figure 1 one negative example is generated for each positive one by changing the target entity while keeping the range of the relation.

2.4 Triple classification

The result of this step is a set of classifications per technique and triple, which may be a binary result (usually 1 or -1), or a probabilistic score. Some techniques do not explicitly classify triples, but output a set of new triples after learning from the training set, or output a source/target node for a query such as <John-S, father-of, ?>. In these cases, there is also a latent classification: the former classifies any triple that is not part of the output as negative, while the later classifies the triples in a group obtained by the query (e.g., all triples with John-S as source and father-of as relation).

Since a knowledge graph may contain thousands of relations, the evaluation of techniques is usually limited to the few ones that are considered specially frequent or relevant [8, 11, 21].

The **inputs** of this step are the training and testing sets, the set of techniques to be compared, and the set of relations to test. The **output** is the set of results from applying each technique to each edge of the input relations in the testing set. In the **example** of Figure 1 four techniques are compared, and only two of the three existing relations are evaluated.

2.5 Statistical analysis

The goal of this step is to generate a report that summarises the results obtained in the classification step with metrics used to evaluate each technique.

The results of the classification can be used to generate a confusion matrix for each evaluated relation, obtaining metrics like precision or recall. The metrics are usually focused on precision [8, 11], since the added knowledge, even if it is a small amount, should be reliable. A proposal with high recall but poor precision results in a high number of false positives that have to be manually checked and removed, defeating the purpose of automated graph refinement.

The difference between techniques regarding each metric is assessed with significance tests, usually paired ones (the observations of a sample can be paired with those of another sample) computed from the metrics of each relation [8, 11].

The **input** of this step is the set of classification results from each technique. The output is a report with metrics and a comparison of the techniques. In the **example** of Figure 1 precision, recall, and F_1 are measured for each relation, and a significance test is used to compare the differences between techniques.

3 Related datasets

Our study of the literature reveals that there is no consensus when it comes to evaluating graph completion proposals. Next, we describe several evaluation datasets in a non-exhaustive study to show that there is variability and what the popular choices are in the workflow. Table 3 summarises our findings.

Table 1. Summary of related datasets.

Proposal	Original graphs	Preprocessing	Splitting	Negatives generation	Triple Classification	Statistical analysis
Socher et al. (2013)	Freebase and WordNet.	Freebase: 13 relations. Wordnet: 11 relations.	~10% for testing. Filtering of "trivial triplets".	1 per positive. Only in Freebase the range is kept.	7 Freebase relations and all WordNet ones are tested.	Accuracy per relation. Averaged in techniques comparison. No sign. tests.
Gardner and Mitchell (2015)	Freebase and NELL.	NELL: some handpicked relations were removed.	25% for testing.	Several per positive. PPR used for selection of new source/target.	24 Freebase and 10 NELL relations are tested.	MAP, MRR, average precision. A paired permutation test measures significance.
Ji et al. (2015)	FB13, WN11, and FB15K	None	FB13, WN11: same as Socher et al. FB15K: 20% for testing.	Generated in the same way as Socher et al.	FB13 and WN11: same as Socher et al. FB15K: all relations are tested.	Accuracy per relation. No sign. tests (only report on their proposal).
Mazumder and Liu (2017)	FB15K, WN18, and ConceptNet	FB15K: 1000 triples from 25 relations. ConceptNet: relations >1000	20% for testing.	4 per positive. 2 change the source, 2 change the target. Keeping the domain/range.	All relations are tested.	MAP and averaged F1.A paired t-test measures significance.

Socher et al. [21] evaluate their proposal using Freebase [5] and WordNet [13] as their original graphs. Preprocessing removes all but 13 relations from the People domain in Freebase, and all but 11 in WordNet. During splitting, they take around 10% of the edges for testing, and remove what they call "trivial test tuples", described as tuples from the testing set in which either or both of their two entities also appear in the training set in a different relation or order.

They generate one negative per positive in the testing set by switching the target entity of the positive. Regarding Freebase, they keep the domain/range of the relation, but since in WordNet all entities share the same type (word), all entities are actual potential candidates. During triple classification, they only test 7 relations from Freebase and all relations from WordNet. They measure the accuracy per relation of their proposal, and the average across relations when comparing several proposals, without using significance tests.

Gardner and Mitchell [8] use Freebase and NELL as their original knowledge graphs. During preprocessing they remove some Freebase relations considered too specific or unhelpful. During splitting, they take 25% of the edges for testing. They do not mention how many negatives are generated, but the datasets they provide show a variable number, usually more than 5 negatives per positive in both training and testing sets. They generate them by changing the source and target in each positive, keeping the domain and range, and weighting candidates by personalised page rank to favour nearby entities. Testing is limited to 25 relations from Freebase (random ones with a number of instances between 1000 and 10000, excluding those with Freebase's mediators [3]), and 10 relations from NELL (the ones with highest frequency and reasonable precision). They measure mean average precision (MAP) and mean reciprocal rank (MRR), using an undisclosed paired permutation test to measure significance.

Ji et al. [9] use Socher et al. [21]'s datasets (FB13 and WN11) as the original graphs, adding Freebase 15K (FB15K) [7]. There is no additional preprocessing. Splitting is the same as the original datasets, which in the case of FB15K is 20% for testing. Since FB15K does not include negative examples in the testing set, they are generated in the same way as Socher et al. The relations used for testing are the same as Socher et al. [21], and all relations for FB15K. They measure the accuracy per relation, without using significance tests, since they only report on the results of their proposal.

Mazumder and Liu [11] use a subset of WordNet with only 18 relations (WN18) [6], FB15K, and ConceptNet [22] as original graphs. During preprocessing, they keep from FB15K 1000 triples from each of 25 randomly selected relations with more than 1000 instances, all triples from WN18, and all triples from relations with more than 1000 instances from ConceptNet (18 in total). During splitting, they take 20% of the edges for testing. They generate four negatives per positive in both training and testing sets. Two are generated by changing the source, and other two by changing the target, both keeping the domain/range of the relation. Testing includes all relations. They measure MAP and averaged F_1 across relations, using a paired t-test to measure significance.

4 AYNEC-DataGen

AYNEC-DataGen, our datasets generation tool, implements the first three steps of the workflow in Figure 1 with several variation points.

Next we describe how our tool implements the aforementioned steps of the workflow, presenting the variation points (VP) that can be configured.

4.1 Preprocessing

AYNEC-DataGen takes as input a file with the input graph. Note that the most popular graphs we have identified (Freebase, WordNet, NELL) do not include literals, but use the additional nodes we mentioned in Section 1.

VP1. Fraction of the graph: The original knowledge graph can be read entirely or only a fraction of it. Each triple has a configurable probability of being ignored, which can be useful to generate reduced versions of large knowledge graphs.
VP2. Relation frequency threshold: Relations with a frequency below a given threshold can be removed. This is useful to remove very specific relations that may be problematic.
VP3. Relation accumulated fraction threshold: A fraction can be set so that only the most frequent relations that cover that fraction of the edges are retained, removing the rest. This is useful to remove large amounts of relations with low frequency while keeping most of the graph intact.
VP4. Inverse removal: Relations r_1 and r_2 are inverses of each other if for each instance of r_1, there is an instance of r_2 with swapped source and target, and vice versa. If this option is toggled, one of the relations in each pair of inverses is removed. This reduces the size of the datasets without removing actual information, and avoids situations where triples are trivially classified as true merely by checking that the inverse relation exists between the entities, which may happen with some datasets [8, 24].

4.2 Splitting

The triples resulting from preprocessing are split into the training and testing sets.

VP5. Testing fraction: The graph can be split using a fraction that is applied to every relation, so that said fraction is taken from the triples of each relation for testing.
VP6. Testing fraction per relation: The graph can be split using a fraction for each existing relation, so that, for each relation, the specified fraction is taken from its triples for testing. This enables full control of the representation of each relation in the testing set.

4.3 Negatives generation

Once the graph is split into two sets of positive examples, we generate negative examples for each positive one.

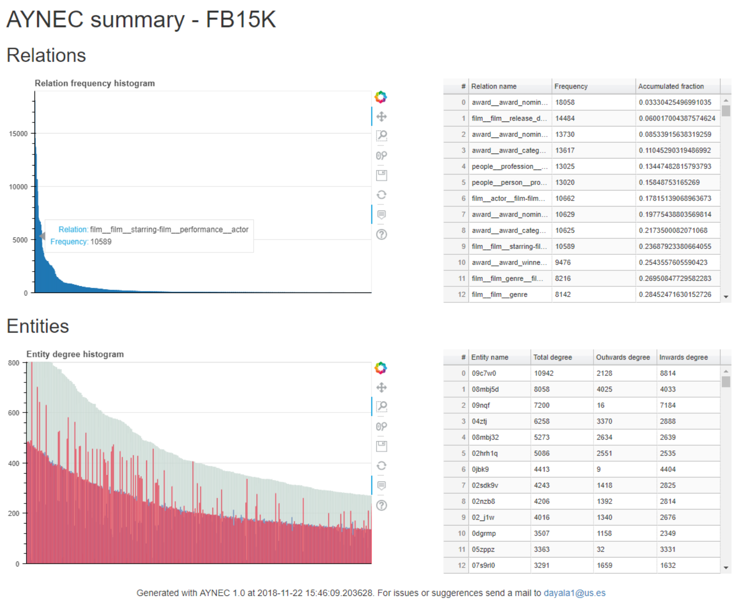
VP7. Training negatives: Negatives can be generated or not for the training set, depending on whether or not techniques are expected to generate their own negatives.

VP8. Negatives per positive: the number of negatives to generate per positive can be a real number. The decimals represent the probability of generating an extra negative example. For example, 2.4 negatives per positive implies that, for each positive, 2 negatives will be generated, with a probability of 0.4 of generating a third one.
VP9. Generation strategy: the generation of negatives is modular, so that several strategies can be chosen and new ones can be easily created. We have implemented the following strategies:

- Changing the source and/or the target of the triple with all entities as candidates [9, 21].
- Changing the source and/or target of the triple with candidates that keep the domain/range of the relation [9, 11, 21].
- Changing the source and target with candidates that keep the domain/range of the relation while weighting by PPR [8].

4.4 Output

The main output of AYNEC-DataGen are two files, "train.txt" and "test.txt", each of which contains a triple per line, and a label which can be "1" or "-1" depending on whether it is a positive or a negative example. Additionally, we generate the following items:


Relation frequency histogram
15000
10000
5000
0
Relation: film__film__starring-film__performance__actor
Frequency: 10589
Entity degree histogram
800
600
400
200
0

AYNEC summary - FB15K

Relations

#	Relation name	Frequency	Accumulated fraction
0	award__award_nomin…	18058	0.0333042549699l035
1	film__film__release_d…	14484	0.060017004387574624
2	award__award_nomin…	13730	0.08533915638319259
3	award__award_categ…	13617	0.11045290319486992
4	people__profession__…	13025	0.13447482815793793
5	people__person__pro…	13020	0.15848753165269
6	film__actor__film-film…	10662	0.17815139068963673
7	award__award_nomin…	10629	0.19775438803569814
8	award__award_categ…	10625	0.2173500082071068
9	film__film__starring-fil…	10589	0.23687923380664055
10	award__award_winne…	9476	0.2543557605590423
11	film__film_genre__fil…	8216	0.26950847729582283
12	film__film__genre	8142	0.28452471630152726

Entities

#	Entity name	Total degree	Outwards degree	Inwards degree
0	09c7w0	10942	2128	8814
1	08mbj5d	8058	4025	4033
2	09nqf	7200	16	7184
3	04ztj	6258	3370	2888
4	08mbj32	5273	2634	2639
5	02hrh1q	5086	2551	2535
6	0jbk9	4413	9	4404
7	02sdk9v	4243	1418	2825
8	02nzb8	4206	1392	2814
9	02_j1w	4016	1340	2676
10	0dgrmp	3507	1158	2349
11	05zppz	3363	32	3331
12	07s9rl0	3291	1659	1632

Generated with AYNEC 1.0 at 2018-11-22 15:46:09.203628. For issues or suggerences send a mail to dayala1@us.es

Fig. 2. Visual summary example.

1. Files listing the relations and entities in the graph. Relations are sorted by their frequency, included in the file. Entities are sorted by their total degree in the original graph, included along with the outward and inward degrees.
2. An interactive visual summary with the aforementioned frequencies and degrees, as depicted in Figure 2 which shows the tables and plots in the file.
3. A file with each identified pair of inverse relations.
4. A file in gexf format with the entire dataset, including the negatives and positives of both training and testing sets. This open format enables to import the dataset in visualisation tools such as Gephi[3]. This is important when developing a completion technique, since it allows the visual study of the topology of the graph and its relations. For example Figure 3 shows a visual representation of positives in WN11-A (one of our generated datasets) where the training and testing sets have a similar topology.

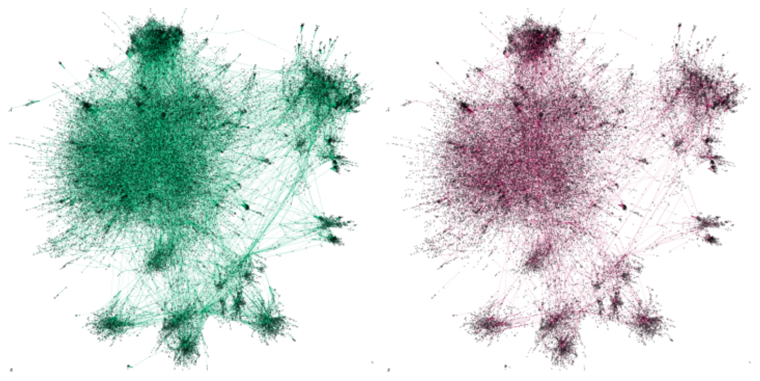

(a) Training (b) Testing

Fig. 3. Visual representation of positives in WN11-A with Gephi.

5 AYNEC-ResTest

AYNEC-ResTest takes the results of several techniques, and computes metrics for each technique and pairwise comparisons using significance tests.

The input of AYNEC-ResTest is a file with the classification results of each technique when applied to every triple. The result of a technique can be binary or a probabilistic score.

AYNEC-ResTest computes, for each technique and relation, the confusion matrix. These matrices are used to compute metrics per technique and relation: precision, recall, and F_1. We consider these metrics to be the most adequate

[3] https://gephi.org/

ones, especially precision. We also compute ranking-based metrics: MAP and MRR, which are also popular [8, 11], but are only adequate when the input of the techniques is a query, and the output a ranking of potential results sorted by score [12].

In addition to the per-relation metrics, our tool computes the macro-average and micro-average of each metric. The macro-average of a metric is computed by averaging the metric of each relation, while the micro-average is computed from the sum of the confusion matrices of each relation. The macro-average is less influenced than the micro-average by unbalanced relation frequencies.

Finally, since the output of a technique can be a probabilistic score, AYNEC-ResTest offers the possibility of computing all the former metrics for different values of the score threshold. The results are, consequently, computed for each relation, for each threshold, and for each technique, allowing the user to compute other metrics related to the precision-recall curve.

Regarding significance, our tool computes the paired, non-parametric Wilcoxon signed rank test [25] to test distribution equivalence, by checking whether or not we can reject the null hypothesis that the difference between each pair of observations (each observation being the value of a metric for a relation) follows a symmetrical distribution around 0. Sometimes, however, due to the behaviour of a technique, some metric values may be missing (for example, it is impossible to compute precision if there are no true or false positives for a relation), which makes it impossible to apply paired tests. To cover this situation, our tool also computes the non-paired Kolmogorov-Smirnov test [25], which is sensitive not only to differences in the median of the distributions, but also to any difference in shape. These tests are computed for each threshold, each metric, and each pair of techniques.

6 AYNEC-Datasets

We have used AYNEC-DataGen to generate specific datasets as a standard evaluation set that we intend to maintain in the future if we identify convenient new configurations or graphs. Regarding the original knowledge graphs they are generated from, we have reused existing high quality resources, some of them from the related datasets we described in Section 3. We have generated our datasets from the following knowledge graphs:

1. **WN18 [6]:** a subset of the WordNet dataset with 18 relations, after filtering the entities that appear in less than 15 triples.
2. **WN11 [21]:** a subset of the WordNet dataset with 11 relations. We only take the positive examples.
3. **FB13 [21]:** a subset of the Freebase dataset with 13 relations from the People domain. We only take the positive examples.
4. **FB15K [7]:** a subset of the Freebase dataset with almost 15000 entities, after filtering those that are not present in the Wikilinks database [20] and appear in less than 100 triples. Some inverse relations are also removed.

5. **NELL [14]:** a knowledge graph crawled from the Web. It is a particularly noisy graph [17].

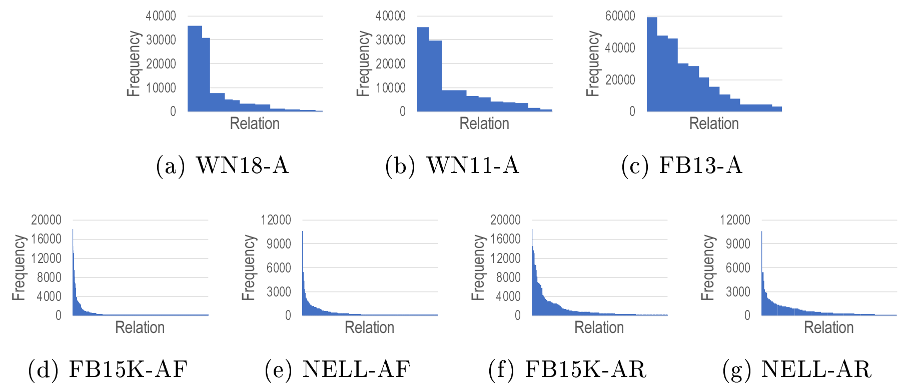
Table 2. AYNEC-datasets.

Original graph	Preprocessing				Splitting		Negatives generation			# of relations	# of entities	# of triples (+/-)	AYNEC-ID
	VP1	VP2	VP3	VP4	VP5	VP6	VP7	VP8	VP9				
WN18	100%	Freq > 1	All	Keep inverses	20% for test	N/A	Training and testing	1	Replace target (random)	18	40943	292884	**WN18-A**
WN11	100%	Freq > 1	All	Keep inverses	20% for test	N/A	Training and testing	1	Replace target (random)	11	38195	220724	**WN11-A**
FB13	100%	Freq > 1	All	Keep inverses	20% for test	N/A	Training and testing	1	Replace target (keep range)	13	74998	570403	**FB13-A**
FB15K	100%	Freq > 1	All	Keep inverses	20% for test	N/A	Training and testing	1	Replace target (keep range)	1219	14951	1075216	**FB15K-AF**
NELL	100%	Freq > 1	All	Keep inverses	20% for test	N/A	Training and testing	1	Replace target (keep range)	515	53934	403734	**NELL-AF**
FB15K	100%	Freq > 1	Accumulate 95%	Remove inverses	20% for test	N/A	Training and testing	1	Replace target (keep range)	341	14951	1022541	**FB15K-AR**
NELL	100%	Freq > 1	Accumulate 95%	Remove inverses	20% for test	N/A	Training and testing	1	Replace target (keep range)	148	53934	217296	**NELL-AR**

After feeding them to our tool, we generated a total of 7 evaluation datasets as depicted in Table 2, which shows the choices regarding every variation point in AYNEC-DataGen. The rationale behind each choice is as follows:

VP1. Fraction of the graph: There is no random filtering of the triples in the datasets, since their size is manageable.
VP2. Relation frequency threshold: We removed relations with one instance, since we cannot include them in both training and testing sets.
VP3. Relation accumulated fraction threshold: Since FB15K and NELL have a large amount of low frequency relations (long tail), we created, apart from datasets with the full set of relations (FB15K-AF and NELL-AF), reduced datasets that only keep 95% of the triples (FB15K-AR and NELL-AR), greatly reducing the number of relations in both cases.
VP4. Inverse removal: We removed inverses in FB15K-AR and NELL-AR.
VP5. Testing fraction: We took 20% of the triples for testing, in line with the related datasets.
VP6. Testing fraction per relation: We took the same fraction from every relation, since we do not focus on some relations in particular that need greater representation.
VP7. Training negatives: We generated negatives for both the training and the testing sets, in order to ease the training of techniques.
VP8. Negatives per positive: We generated one negative per positive, which is the most frequent amount in the related datasets.
VP9. Generation strategy: We generated negatives by changing the target of each positive example, since graphs are completed by applying the classifier of a relation to every possible target of one source entity, and this generation strategy creates the most similar scenario. In most datasets, we kept the range of each relation by using as candidates entities that appear as target in another triple of the same relation. However, since all entities

in WordNet share the same nature (they are all words), all entities are candidates for all relations. The strategy by Gardner and Mitchell [8] is more complex, but it has not been assessed or discussed whether or not it makes the evaluation better.



(a) WN18-A (b) WN11-A (c) FB13-A

(d) FB15K-AF (e) NELL-AF (f) FB15K-AR (g) NELL-AR

Fig. 4. Relation frequency histogram, with relations sorted by frequency.

Figure 4 shows several plots with the frequencies of the relations in each dataset. FB15K-AR and NELL-AR reduce the long tail by trimming the less frequent relations. The minimum relation frequency in FB15K-AF is 2, while in FB15K-AR it is 153. Similarly, the minimum relation frequency in NELL-AF is 3, while in NELL-AR it is 120.

Compared to the related datasets [8, 9, 11, 21], ours include more knowledge graphs, they solve some existing problems like the presence of inverses or low frequency relations, they follow similar strategies when it comes to parameters like the testing fraction or negatives generation strategy, they contain meta-information about relations and entities, and they are presented in a format that makes it easy to import them into graph visualisation tools.

7 Conclusions

In this paper, we have presented a new suite for the evaluation of knowledge graph completion techniques. In the literature, each proposal is evaluated with a different setup and using different metrics and significance tests, which motivated the creation of an unified suite that streamlines evaluation.

The tools and datasets of our suite are customisable so that they adapt to a variety of scenarios, in case a different configuration is needed beyond the original ones. The source code of the tools is documented, and uses popular input/output formats in order to ease its adoption by researchers.

The generated datasets follow what we consider to be the most interesting strategies for homogeneous out-of-the-box evaluation, reusing popular subgraphs with useful preprocessing. The metrics and significance tests best suited for the datasets are implemented in AYNEC-ResTest, which takes care of the comparative analysis of techniques from a set of results.

The tools and datasets are publicly available online. We intend to maintain and expand them if new requirements are identified, such as novel negatives generation strategies or new knowledge graphs with interesting properties.

Acknowledgements Our work was supported the Spanish R&D&I programme by grant TIN2016-75394-R. We would also like to thank Prof. Dr. José Luis Ruiz-Reina, head of the Computer Science and Artificial Intelligence Department at the University of Seville, who kindly provided us with the invaluable resources that helped us in our research.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: Dbpedia: A nucleus for a web of open data. In: 6th International Semantic Web Conference, pp. 722–735 (2007), https://doi.org/10.1007/978-3-540-76298-0_52
2. Ayala, D., Hernández, I., Ruiz, D., Toro, M.: Tapon: A two-phase machine learning approach for semantic labelling. Knowledge-Based Systems (2018)
3. Bast, H., Bäurle, F., Buchhold, B., Haußmann, E.: Easy access to the freebase dataset. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 95–98, ACM (2014)
4. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. **5**(3), 1–22 (2009), https://doi.org/10.4018/jswis.2009081901
5. Bollacker, K.D., Cook, R.P., Tufts, P.: Freebase: A shared database of structured general human knowledge. In: AAAI 22, pp. 1962–1963 (2007)
6. Bordes, A., Glorot, X., Weston, J., Bengio, Y.: A semantic matching energy function for learning with multi-relational data - application to word-sense disambiguation. Machine Learning **94**(2), 233–259 (2014), https://doi.org/10.1007/s10994-013-5363-6
7. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems, pp. 2787–2795 (2013)
8. Gardner, M., Mitchell, T.M.: Efficient and expressive knowledge base completion using subgraph feature extraction. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1488–1498 (2015)
9. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, pp. 687–696 (2015), https://doi.org/10.3115/v1/P15-1067

10. Junghanns, M., Kießling, M., Teichmann, N., Gómez, K., Petermann, A., Rahm, E.: Declarative and distributed graph analytics with GRADOOP. PVLDB **11**(12), 2006–2009 (2018)
11. Mazumder, S., Liu, B.: Context-aware path ranking for knowledge base completion. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 1195–1201 (2017), https://doi.org/10.24963/ijcai.2017/166
12. McFee, B., Lanckriet, G.R.: Metric learning to rank. In: Proceedings of the 27th International Conference on Machine Learning, pp. 775–782 (2010)
13. Miller, G.A.: Wordnet: A lexical database for english. Commun. ACM **38**(11), 39–41 (1995), https://doi.org/10.1145/219717.219748
14. Mitchell, T.M., Cohen, W.W., Jr., E.R.H., Talukdar, P.P., Yang, B., Betteridge, J., Carlson, A., Mishra, B.D., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E.A., Ritter, A., Samadi, M., Settles, B., Wang, R.C., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., Welling, J.: Never-ending learning. Commun. ACM **61**(5), 103–115 (2018), https://doi.org/10.1145/3191513
15. Pasca, M., Lin, D., Bigham, J., Lifchits, A., Jain, A.: Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In: AAAI, pp. 1400–1405 (2006)
16. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web **8**(3), 489–508 (2017)
17. Paulheim, H., Bizer, C.: Improving the quality of linked data using statistical distributions. Int. J. Semantic Web Inf. Syst. **10**(2), 63–86 (2014), https://doi.org/10.4018/ijswis.2014040104
18. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: The Semantic Web Conf. (ESWC), pp. 593–607, Springer (2018)
19. Shao, B., Wang, H., Li, Y.: The trinity graph engine. Microsoft Research **54** (2012)
20. Singh, S., Subramanya, A., Pereira, F., McCallum, A.: Wikilinks: A large-scale cross-document coreference corpus labeled via links to wikipedia. University of Massachusetts, Amherst, Tech. Rep. UM-CS-2012 **15** (2012)
21. Socher, R., Chen, D., Manning, C.D., Ng, A.Y.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in neural information processing systems, pp. 926–934 (2013)
22. Speer, R., Havasi, C.: Representing general relational knowledge in conceptnet 5. In: LREC, pp. 3679–3686 (2012)
23. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW 2007, pp. 697–706 (2007), https://doi.org/10.1145/1242572.1242667
24. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Work. Continuous Vector Space Models and their Compositionality, pp. 57–66 (2015)
25. Woolson, R.: Wilcoxon signed-rank test. Wiley encyclopedia of clinical trials pp. 1–3 (2007)