Improving Link Prediction in Multi-Relational Knowledge Bases

Through Natural Language Processing

By

PAUL URSULEAN

Supervisor

JACOPO URBANI

Department of Computer Science

VRIJE UNIVERSITEIT AMSTERDAM

JULY 2018

ABSTRACT

Link prediction in the context of multi-relational data describes the process of extrapolating relationships which are not explicitly known from patterns in the known data, which can be defined in a multitude of ways. Due to the large scale and inherent incompleteness of this type of data, link prediction is important for completion, analysis and error-correction. This research builds on already existing methods by leveraging natural language found in the data to achieve improved results under the right circumstances. This paper contributes a proof-of-concept, and aims to serve as a jumping-off point for further research into the topic.

I. INTRODUCTION

The problem of encoding human knowledge into a format understood by computers has been pondered for almost as long as computers have existed. The essential human desire for self-understanding will hopefully culminate in the ability to artificially replicate intelligence itself. While the use of technology for information storage is ubiquitous, this is a far cry from capturing anything resembling human intelligence. A proposed solution to bridge this gap relies on knowledge bases as a means to store complex information. In contrast to the simple, linear nature of traditional databases, knowledge bases can be abstracted to graph-like models of entities and relationships, or multi-relational data. These models have come into widespread use in search engines, social networks, E-commerce and entertainment.

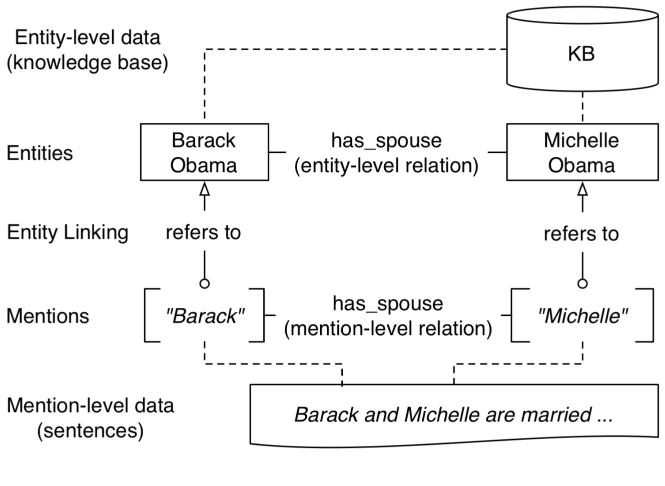

Fig. 1. Common KB Construction Pipeline

A. Multi-Relational Data and Knowledge Bases

Multi-relational data is analogous to a graph whose nodes correspond to *entities*, and whose edges correspond to directed *relationships* between said entities. This graph can be represented in an adjacency list-like format, by triples of the form (*head, relationship, tail*), or in short (*h, r, t*).

This paper concerns itself with knowledge bases, which store general facts about the world as multi-relational data. In comparison with the rigidity of databases, the flexibility of this format allows a storage of knowledge more in line with the human way of thinking about the world: in general terms, with *entities* and *relationships* that can take form at differing levels of abstraction.

In their original form, knowledge bases were one half of *knowledge-based systems*, the other half being an *inference engine*, which would apply rules to deduce new facts and find inconsistencies.[1]

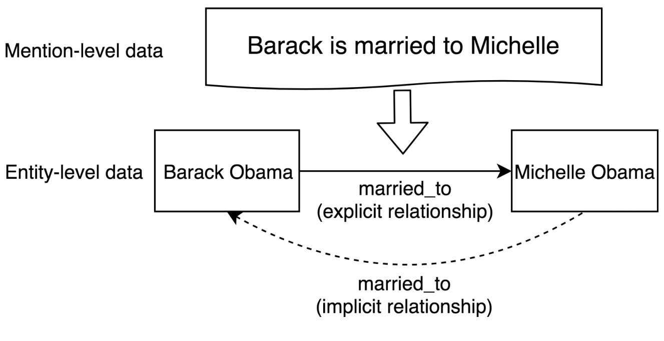

Fig. 2. Link Prediction Example: By extrapolating the bi-directional pattern of the *married_to* relationship, the triple *(Michelle Obama, married_to, Barack Obama)* can be predicted from the explicitly known mirrored triple *(Barack Obama, married_to, Michelle Obama)*

While the distinction between knowledge bases and databases has become less clear since, this definition of knowledge-based systems captures the task of link prediction succinctly.

B. Motivation

The need for effective link prediction comes from the inherent scaling problems that affect large knowledge bases such as Freebase,[2] Google Knowledge Graph,[3] and YAGO.[4] Since the ultimate goal is to compile the entirety of human knowledge into a homogeneous format, the task obviously demands automation, which comes with its share of problems.

Firstly, information in general is not evenly distributed across entities. The Pareto distribution class models situations of exponential inequality, wherein, for example 80% of the effects are caused by 20% of the causes. This distribution is the source of the Pareto principle, or the 80/20 rule, which has been shown empirically to hold true in many aspects relating to business, socio-economics and software.[5] It is immediately obvious that the distribution of publicly available information would follow some variant of the power law distribution, with a large portion of the information in the public domain relating to a relatively small share of entities. This is not inherently bad, but for the purposes of creating a large-scale knowledge base, the dataset is incomplete, with a large amount of variance in the amount of information available for potentially similar entities.

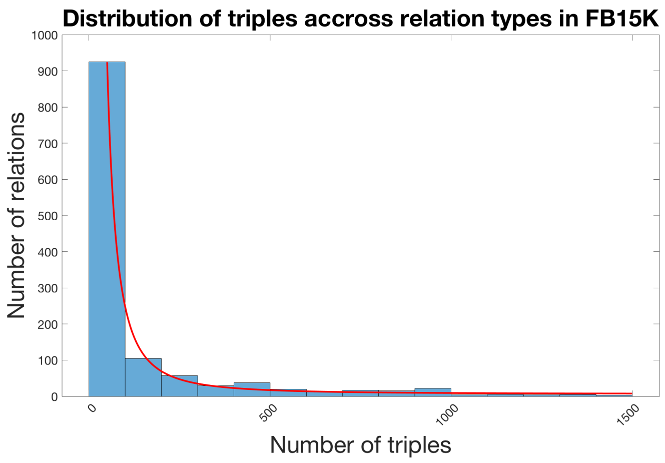

Fig. 3. It is observed that the distribution of triples across relationship types in knowledge bases also seems to follow a variant of the power law distribution.

Furthermore, misinformation on the Internet has always been a problem, but has become especially rampant in recent years. Due to the metrics used to assign importance to stories on various platforms and search engines, and also due to human nature, it has been shown that false information spreads faster than the truth, with true stories taking on average 6 times longer to reach 1500 people.[6]

Knowledge bases often suffer from incomplete and incorrect information, especially as their scope broadens to general facts about the world.[7] Link prediction addresses both of these problems to a degree, although it also suffers from them somewhat. Since the technique is always based on patterns no matter the approach, it may suffer from the inconsistent distribution of information and fail to predict a pattern where there may be one, due to the absence of its precursors in the knowledge base. In addition, drawing conclusions from false premises is obviously undesirable. It is worth noting however that link prediction should not be the first line of defense against these problems, and especially against misinformation. Ideally, the *knowledge base construction* stage would filter out most of the noise, and do a good job of capturing known facts, allowing the link prediction process to focus on finding the unknown ones.

C. Related Work

This paper primarily intends to build on TransE, a method which learns latent representations for entities and relationships in the form of vectors in low-dimensional space, with the property that relationships from one entity to another can be described by vector translation.[8] In other words, it holds that $head + rel \approx tail$, given embeddings *head, rel, tail* for an existing triple inside the knowledge base. TransE leverages the global and local connectivity patterns of the graph in order to produce state-of-the-art results in link prediction on multiple knowledge bases, while remaining relatively scalable and easy to train. The method is at its best in the dense parts of the graph, however it leaves something to be desired in the sparse sections. While this results in a remarkable average performance, there is room for improvement, especially if the focus is graph completion, in which case its performance on sparse sections of the graph becomes increasingly important.

Thus, the objective of this research is to show an improvement in link prediction when considering sparse sections of the graph. To this end, the viability of incorporating natural language processing into the task is explored. Given that a knowledge base contains facts about the world by definition, the data stored inside it must correspond with natural language data at some level. The hypothesis states that this natural language data contains information which is not currently being leveraged, that can potentially aid in prediction.

In Section II, the process of extracting natural language textual data is discussed, and subsequently linked to the possible machine learning approaches given the format, in section III. Section IV shows a comparison of link prediction accuracy between TransE and the new approach in various circumstances in the context of the FB15K dataset, and section V discusses efficiency and scalability. Finally, the conclusion in section VI lays out possible future research directions.

II. TEXT EXTRACTION

It is important to evaluate the format of the knowledge base (KB) in order to determine the nature of the data that can be used in natural language processing tasks. Unfortunately this depends somewhat on the specifics of the particular KB, however an effort was made to keep the solution relatively general by focusing on elements that should be commonly found in KBs. Nonetheless there is some work required in order to migrate the solution to other KBs, as they all have their own conventions.

A. Labels

The most basic and standard piece of textual information tied to an entity is naturally its label, a general term for a title or name. The label is also a property shared by the relationships, and thus simple sentences of the form (*subject, predicate, object)* can be formed. Unfortunately, a few problems arise when taking this approach.

First and foremost, we run into a problem of specificity. In order to extrapolate patterns, a certain level of generality is required such that commonalities may be identified. Reducing an entity to its exact name only allows extrapolation to entities with the same or similar name, depending on the method used. Evidently this approach would not work due to the fact that a correlation of labels does not promise any other meaningful similarities. To illustrate this fact, it is enough to point out that two persons sharing the same name are not required to have anything in common.

In essence, the issue is that the label by itself does not provide enough information about an entity. If we chose to focus on the relationships around an entity and drew similarities with other entities with similar relationship structures, we would merely be taking the same approach as TransE, only without the advantages that come with using low-dimensional vector spaces. Thus, some additional information needs to be extracted from the textual label data.

Named Entity Recognition is a common task in natural language processing, which extracts relevant entities from a piece of text and classifies them into broad categories.[9] This is a useful technique that can abstract entity labels into more general terms, which can then be correlated with other entities sharing the same classification.

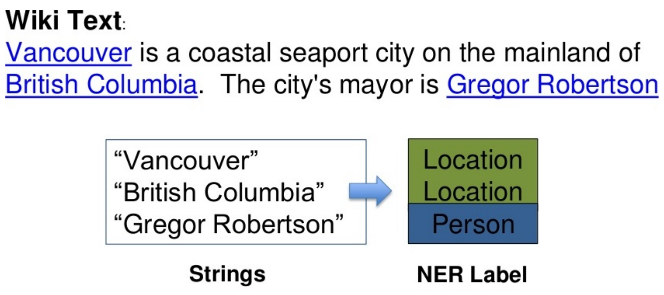
Wiki Text:
Vancouver is a coastal seaport city on the mainland of British Columbia. The city's mayor is Gregor Robertson
"Vancouver"
"British Columbia"
"Gregor Robertson"
Location
Location
Person
Strings
NER Label

Fig. 4. Example of Named Entity Recognition

Unfortunately, in practice not all entities could be resolved to a consistent category. The most common reason for failure of classification was the label referring to an abstract concept, such as an event or movie title, where the Named Entity Recognizer was attempting to derive a literal meaning from each word. This is an obvious limitation of considering only entity labels, which often do not contain information about what that entity actually is, especially when the label has a figurative meaning.

Empirical testing showed that only around 60% of the entities were able to be classified. To make matters worse, triples contain two entities, thus the

TABLE I

Example Unresolved Entity Labels

Unresolved Entity	Actual Description
Hope and Glory	Film
The English Patient	Film
Soul music	Music genre
2008 Summer Olympics	Sports event
Eagles	Band
Academy Award for Best Actor	Award

TABLE II

Example First Sentence Descriptions

Entity	First Sentence Description
Fargo	Fargo is the largest city in the State of North Dakota, accounting for nearly 16% of the state population.
Russian	Russian is an East Slavic language and an official language in Russia, Belarus, Kazakhstan, and Kyrgyzstan
Restaurant	A restaurant is a business which prepares and serves food and drinks to customers in exchange for money.
Park Chu-young	Park Chu-young is a South Korean footballer who is currently playing for FC Seoul

portion of triples which could be resolved into sentences fell to around 30%. Lastly, in practice the categories were found to be too broad and generic to offer any meaningful information.

A valuable insight gained from this experiment is that there is a delicate balance on the spectrum of specificity. In the case of exact names, it was apparent that context is important, and a simple name or title often does not provide enough meaningful information to be of use. However successful classifications of the Named Entity Recognizer showed that it is possible to abstract until the entity possesses almost no distinguishable qualities, in which case drawing correlations is also useless, as the valuable information is drowned out by noise.

B. *Descriptions*

Entity descriptions were found for all entities present in FB15K, and are fairly standard attributes for large knowledge bases. While the length of descriptions varies, an important observation is that the first sentence always directly describes the entity without assuming any prior knowledge. This makes sense, since it is meant as an introduction for readers who are not at all familiar with that entity, but it also happens to align with previous insights about the type of information that is valuable for the purposes of this research. The descriptions provide context while also remaining specific enough to extract the distinguishable properties of an entity.

III. MACHINE LEARNING APPROACH

Now that the dataset of first sentence descriptions is decided, the next crucial step lies in choosing an appropriate predictive approach. This measure of appropriateness depends not only on the accuracy of the estimator's results, but also on the ability of these results to be meaningfully compared to the results of TransE.

A. *TransE Accuracy Measure*

Since TransE does not output a decision category or a regression score, measuring accuracy becomes a concern. Naturally, it should be measured by how closely it follows its intended purpose of representing relationships as translations in the vector space from the heads to the tails of their respective triples. However measuring the average distance between a translation and its target will not be of much use since distances in this vector space are not meaningfully analogous to any real-world measure of success. In other words, a pure distance measure is not useful in this scenario because it is only meaningful in the context of this low-dimensional vector space, and comparison with any other model type will not be possible.

Instead, the broader goal of the model should be considered, namely the ability to predict potential links, by showing clear distinctions between probable and improbable ones. This means that what needs to be measured is not absolute distance between approximation and target, but relative distance in comparison with distance to other false targets. This is a measure of the model's ability to distinguish between real and false facts, and can be meaningfully compared with other models.

Thus it makes sense to use a ranking score which ranks all targets in order of their distance to one translation, with the goal of finding the target corresponding to that translation as low on the ranking list as possible. In other words, the goal is for the ranking of each target with respect to its corresponding translation to be as low as possible. A detailed specification for this ranking algorithm is available at Algorithm 1.

```
Data: Three lists of TransE embeddings
      $(emb_{heads}, emb_{rels}, emb_{tails})$,
      representing the set of triples,
      maintaining order such that the
      embeddings for the triple at index $i$ are
      found at
      $(emb_{heads}[i], emb_{rels}[i], emb_{tails}[i])$
Result: Returns average rank for list of
      embedded triples
$n \leftarrow length(heads)$;
$translations \leftarrow emb_{heads} + emb_{tails}$;
$ranks \leftarrow$ array of length n;
for $i \leftarrow 1$ to $n$ do
    $distances \leftarrow$
     $norm(emb_{tails} - translations[i])$;
    // norm returns the L1 norms
       for each vector in the
       list
    $indices \leftarrow argSort(distances)$;
    // argSort returns the
       indices that sort an array
       increasingly.
    $rank \leftarrow indexOf(i, indices)$;
    // indexOf returns the
       location of the first
       parameter in the array
       defined by the second
       parameter.
    $ranks[i] \leftarrow rank$;
end
return $average(ranks)$;
```

Algorithm 1: Compute accuracy for TransE embeddings. Implementation code for this and following algorithms can be found here.

With this ranking taken into account, the natural language model also needs to be able to rank in the same manner, which automatically narrows the search down to regression-based models. Additionally, the model will also be evaluating data at the level of whole triples, just like TransE, in order for an analogous ranking algorithm to be applied.

B. Feature Extraction

Feature extraction is arguably the most important step in the process, as it determines the information that will be available to the model. In addition, it has an influence on model choice itself, due to different model classes benefiting from different features and feature vector types.

With the dataset arrived at in Section II, it makes sense to make use of all the words available for each entity, as the descriptions are very specific to the entity, while not requiring any prior knowledge. As such, each word contains valuable descriptive power. Thus, we assign an individual ID to each unique word found in the set of all entity descriptions. This mapping from words to IDs becomes the *vocabulary*. The feature vector for an entity results from parsing a description, and converting it to a sparse boolean vector wherein only values that correspond with IDs for the parsed words in that specific description are activated. This is called one-hot encoding, or count-vectorization.

In addition to knowing about the presence of words, it is also useful to have insights regarding their significance. Accordingly, we use TF-IDF weighing to give a measure of importance to each word. This technique normalizes words both based on their frequency within a description, or document, and on their inverse frequency across all documents, resulting in a fairly balanced measure of importance for each word.[10] In combination with one-hot encoding, this technique results in each entity represented as a vector in high-dimensional vector space, wherein each dimension corresponds to a particular word from the vocabulary, and the value within that dimension corresponds to its significance if present.

s_1	I enjoy flying.
s_2	I like NLP.
s_3	I like deep learning.

$$\begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{cccccc} \textit{deep} & \textit{enjoy} & \textit{flying} & \textit{learning} & \textit{like} & \textit{nlp} \end{array} \\ \left[\begin{array}{cccccc} 0 & 0.707 & 0.707 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.605 & 0.796 \\ 0.6228 & 0 & 0 & 0.6228 & 0.474 & 0 \end{array}\right] \end{array}$$

Fig. 5. Example TFIDF Feature Extraction

TABLE III

EXAMPLE RELATIONSHIP LABELS

Relationship label
</location/location/people_born_here>
</award/award_winner/awards_won./award/award_honor/award_winner>
</people/person/education./education/education/institution>
</sports/sports_team/roster./sports/sports_team_roster/position>
</award/award_ceremony/awards_presented./award/award_honor/award_winner>
</music/genre/artists>
</sports/sports_team/roster./american_football/football_roster_position/position>

C. Model Choice

Considering each entity corresponds to a sentence, it is easy to see that the number of words in the vocabulary will explode. However it is also important to note that each entity description will only contain a small fraction of the total words in the vocabulary, creating a perfect use case for a sparse-vector representation. For this reason, the choice of model is very important, as the task requires a model class that is able to elegantly handle sparse vectors with thousands of features.

The Linear SVM is a very popular choice for sparse data, and by extension for text-classification tasks, by virtue of its linear scaling with the number of nonzero values in the feature vector.[11] Thus the Linear SVM is a perfect choice for the use-case at hand.

D. Feature Vector Construction

Until now we have only been focusing on entities separately, but as mentioned previously, the natural language model needs to be able to parse entire triples.

Relationships in FB15K do not contain any information outside of their label, which is of a hierarchical structure. While this hierarchical information could be useful for correlation, we chose to ignore it because the natural language data that can be extracted from this format is limited at best. In addition, TransE already does a great job of leveraging the structure of the graph, or the patterns of connectivity for the various types of relationships, which would likely be a large portion of the information captured from these labels.

However we are still interested in capturing the similarity generated by entities which use the same relation, as this is a form of implicit clustering. Thus it is useful to separate triples at the relationship level and train separately for each relation type. This decision helps with scaling, while still making use of the different relationship types, implicitly. However it is important to note that the sacrifice of this decision is the loss of inter-relation correlation of entities. Depending on how dissimilar entities with different relations are, this may or may not be a bad thing, but it is important to take notice of it going forward.

Putting everything together, it is clear that the feature vector still needs to describe two entities, even if the relationship is implicitly known. There are multiple ways of doing this. First, the two sentences corresponding to the entities can be concatenated together and converted into a feature vector of the same length as one made for a single entity, but with less sparsity. This is not a good idea because information about the order of entities, i.e. which one is the *head* and which one is the *tail*, is lost. If two triples both have a word mentioned once, each in different positions, this translates to the same output to the feature vector for the dimension of that word. This contributes ambiguity through loss of meaning.

Alternatively, the two entities can be converted to vectors separately, and concatenated afterwards. This method has the benefit of maintaining order by separating words used in the *head* from words used in the *tail*, but the obvious drawback is the fact that the feature vector will end up being twice as long. This is the method that we will be moving forward with.

Everything is now in place. The pipeline is created with the custom feature extraction process mentioned above, and the Linear SVM as the regression-based estimator.

IV. APPLYING THE MODEL

A. Metrics

At this stage, metrics must be chosen for evaluation, in order to gauge the success of the model. This is important not only as a sanity check, but also as a means to optimize the *hyperparameters* of the whole pipeline. For this purpose three metrics

TABLE IV

HYPERPARAMETER SEARCH RESULTS

Hyperparameter	Selected Value	Percentage of Best Results
min_df	0.005	36.1%
max_df	0.1	27.3%
max_features	None	36.3%
C	0.5	70.1%

are chosen, namely the *Coefficient of Determination* R^2, as well as the *Mean Absolute Error* and *Median Absolute Error*. The former describes the amount of variance in the data that the model accounts for, and caps out at a maximum value of 1. The latter two measure the absolute error. Both mean and median metrics were chosen in order to get an idea if there is a skew in the errors. This is important because the mean is very susceptible to outlier values, while the median is almost entirely unaffected by them.

B. Hyperparameter Selection

It is already known that the Linear SVM model is partial to sparse data, and is often used for text-classification purposes.[11] Thus the hyperparameters of interest in this instance more so concern the TFIDF application, rather than the model itself. As such, we perform a basic grid search with cross validation on common-sense values of the *max_df*, *min_df* and *max_features* TFIDF parameters, which impose various limitations on document frequency for tokens to be admitted into the vocabulary. In the case of *min_df*, it only adds a certain word to the vocabulary if its frequency exceeds a certain threshold. In contrast *max_df* only adds words to the vocabulary if their frequency falls below its threshold. Finally, *max_features* limits the number of words in the allowed in the vocabulary to the parameter value, with tokens ordered by term frequency across the corpus. Experimenting with these values can drastically change the number of features, and can help prune terms which do not contribute any additional predictive power.

In addition, the *C* parameter of the Linear SVM estimator is searched. This parameter determines the penalty for error in regression. Larger values of *C* prefer smaller-margin hyperplanes if it leads to a smaller error, and vice versa.

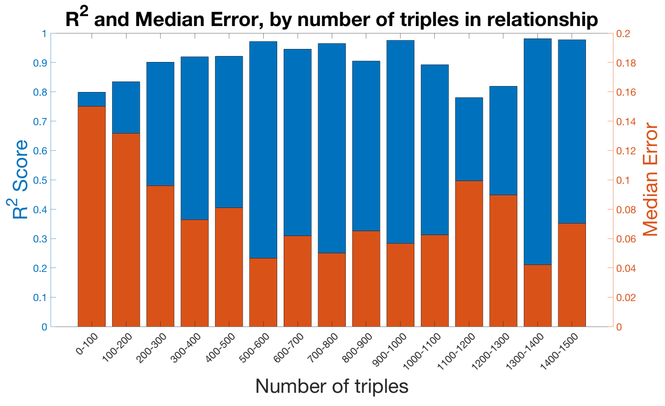

Fig. 6. Sample performance with selected hyperparameters

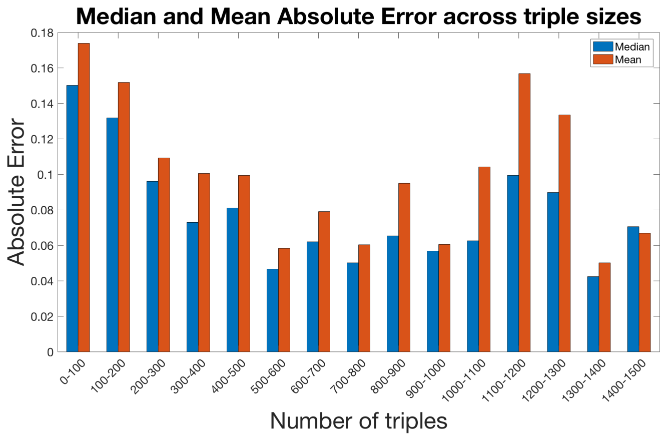

Fig. 7. Sample error analysis for chosen hyperparameters. Higher mean absolute error suggests that there could be a right-skew in the distribution of errors.

C. Training

As mentioned previously, training was performed on a per-relation basis, using a 90% training split. A different pipeline consisting of a vectorizer-estimator pair was thus generated for each relationship and saved into its own separate file. A unique TFIDF vectorizer object needed to be included for each relationship due to the difference in chosen features for each of them.

D. Comparison with TransE

Now that each relationship type has its own model to refer to, all that is left to do is build an environment that simulates the conditions under which TransE is tested as closely as possible. The primary difference between these two testing environments is that instead of performing vector addition on the *head* and *rel* embeddings, the actual relationship ID will determine the model file used. The comparison between the real and fake triples remains the same conceptually, however as far as the implementation goes, these two can vary wildly because of the possible different implementations used for TransE and the natural language model.

```
Data: Three lists of entity and relationship
      IDs $(heads, rels, tails)$, maintaining
      order such that the embeddings for the
      triple at index $i$ are found at
      $(heads[i], rels[i], tails[i])$
Result: Returns average rank for list of
        regular triples
$n \leftarrow length(heads)$;
$ranks \leftarrow$ array of length $n$;
for $i \leftarrow 1$ to $n$ do
    $h_i \leftarrow [[heads[i]] * n$;
    $r_i \leftarrow [[rels[i]] * n$;
    // These operations create
       repeating arrays of length
       n, with values heads[i],
       respectively rels[i]
    $pipeline \leftarrow$
     $load\_model\_from\_file(rels[i])$;
    // load_model_from_file
       fetches the pipeline
       corresponding to the
       parameter given
    $predictions \leftarrow$
     $pipeline.predict(h_i, r_i, tails)$;
    $indices \leftarrow argSort(predictions)$;
    // This time argSort sorts
       decreasingly
    $rank \leftarrow indexOf(i, indices)$;
    $ranks[i] \leftarrow rank$;
end
return $average(ranks)$;
```

Algorithm 2: Compute rankings for natural language model

E. Results

A direct comparison can be made between TransE and the natural language model by running the two algorithms specified in Algorithm 1 and 2 side by side on the same set of triples. The choice of this set of triples then constitutes the circumstances, which can be further analyzed in conjunction with the performance comparison to determine the strengths and weaknesses of the solution developed in this paper.

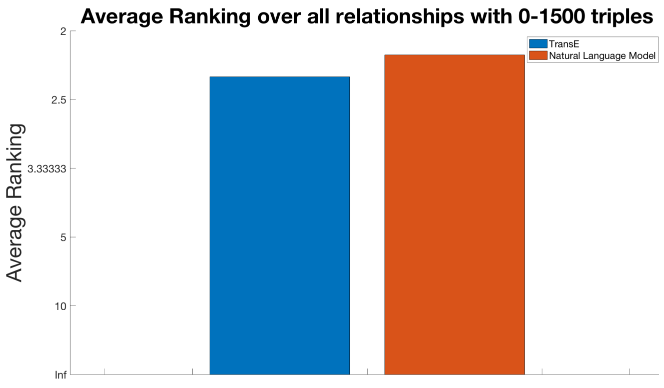

Fig. 8. Comparison of average performance between TransE and the Natural Language Model.

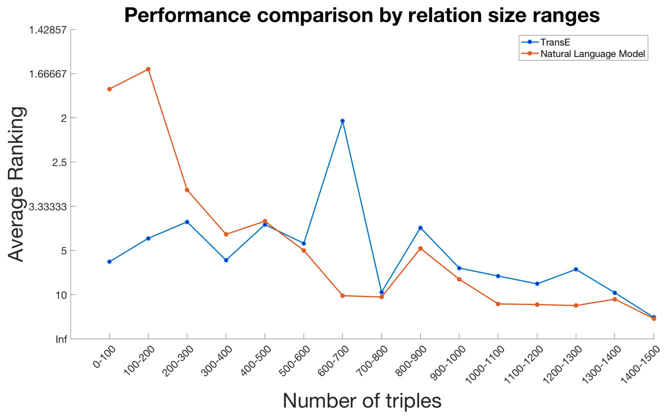

Fig. 9. Comparison of performance between TransE and the Natural Language Model. Note that in rankings, a lower number is better, however in this figure the y-axis is flipped.

While Figure 8 shows a clear improvement in average performance when considering all relationships having up to 1500 triples, Figure 9 shows that this improvement or lack thereof is very much dependent on the number of triples that the involved relationships contain. If the number of triples per relationship is to be taken as a measure of sparsity for that relationship, these results are in line with the hypothesis put forward in Section I: the natural language model excels where TransE does not, namely the sparse areas of the graph.

V. Efficiency and Scalability

Throughout the development of this project, a conscious effort was made to manage the problem of scale inherent to large knowledge bases. This is important because scalability is a cornerstone of TransE, and as such, any proposed improvement should be able to ’keep up’ in terms of time complexity. This section will cover the optimization efforts undergone for this project.

A. Feature Extraction Hyperparameters

As mentioned previously, the feature extraction process defines the data that will be fed to the model. As such, the parameters of the vectorizer will have an effect on the resulting features, and most importantly, on the number of features. Table IV shows that while all vectorizer parameter values were chosen due to their performance, their majorities were not absolute, meaning that in over 50% of cases, other values were optimal. This shows that performance does not necessarily increase with more features, as the chosen values for *min_df* and *max_df* were neither the biggest or smallest tested. This opens up the possibility that adding some features makes prediction more difficult, by adding noise. This phenomenon can be further studied to determine the optimal feature selection process, which can both improve performance and accuracy.

B. LIBLINEAR

Multiple implementations of the Linear SVM were tested for this project, with the LIBLINEAR implementation showing the best efficiency, cutting training time down more than tenfold compared to the other implementations, presumably due to its linear scaling.[11] However this efficiency gain proved to come at a cost, as the results did not fare as well against TransE as with the other implementation. Nevertheless, LIBLINEAR is a more efficient implementation for this use case, and still showed an improvement over TransE under the right circumstances, with the difference in final performance most likely coming down to different default parameters which were overlooked at training time.

C. Libraries

A suspected weak point of this project with respect to performance comes from the fact that the TransE implementation utilized TensorFlow, a lower-level API, while the natural language model

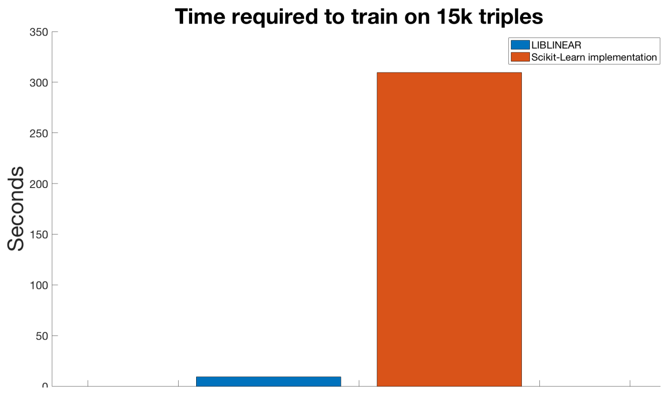

Fig. 10. Comparison of training speed between Linear SVM implementations

was implemented using Scikit-Learn, a higher-level API. We hypothesize that significant performance gains are to be had if the two models are homogenized to a single library.

VI. Conclusion and Future Work

It is important to keep in mind the purpose of this paper, namely to serve as a jumping-off point for future research into this topic. While exploratory in nature, this research proves the possibility of leveraging natural language data for state-of-the-art results in link prediction in select circumstances. Whether this work can be generalized to be applied at the same scale as TransE is a matter of scaling and analysis, and thus the following research directions materialize.

A. Scaling

As hinted at in the previous section, working with the Linear SVM implementation of LIBLINEAR to optimize results seems like the immediate next step, as the performance increase cannot be ignored. This is not an endeavor for a full research paper by itself, however the question of hyperparameter optimization with respect to the ranking results is of note. Until now, hyperparameter optimization was performed using the built-in R^2 metric of the SVR, however the discrepancy in final results between LIBLINEAR and other implementations shows that this metric does not perfectly correlate with ranking results. Thus, further analysis into this matter is warranted, with the solution perhaps being to implement a custom model whose custom performance metric is the ranking performance.

In addition to tweaking the LIBLINEAR implementation of the Linear SVM, the other task which would help with performance is homogenizing the two models into an implementation which only uses one library, as mentioned in the previous section. This would be useful as lower-level APIs are generally more efficient and flexible.

B. Analysis and Model Combination

Another potential research direction is an analysis of the results of this paper. More specifically, it is important to find a pattern in the results of the natural language model in comparison with TransE, such that the right model can be employed in the right situation for the best predictions. While this can be abstracted into a machine learning problem in and of itself, it is also important to find a reasoning behind these results, for the benefit of our understanding of the problem. While the current understanding is that the natural language model performs better on sparse areas of the graph, there are many different measures of sparsity, which could offer additional insight upon analysis.

C. Feature Extraction and Model Choice

While this paper chose to focus on TFIDF feature extraction, this is hardly a restriction. A word embedding scheme such as Word2Vec[12] could prove fruitful, and it would conform to the theme of low-dimensional embeddings. In addition, neural network models for both TFIDF and Word2Vec would be a natural next step.

D. Generalization

The overall significance of the results achieved in this paper depends on their ability to be generalized to a whole knowledge base. While the scaling concern was voiced earlier, there are also other matters related to this generalization. Firstly, there is the question of whether the FB15K dataset constitutes a realistic representation of Freebase, or an idealistic one. The answer to this question heavily influences the significance of the results since if this dataset is an ideal one for the TransE technique, results in practice might be better for the natural language estimator, or vice-versa. Furthermore, there is the concern of porting this solution to other knowledge bases and what that might entail. Since this technique relies on descriptions, results may vary depending on format and conventions of other knowledge bases.

References

[1] Hayes-Roth, Frederick, Donald Arthur. Waterman, and Douglas B. Lenat. *Building Expert Systems*. Reading, MA: Addison-Wesley, 1987.

[2] Bollacker, Kurt, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. "Freebase: A Collaboratively Created Graph Database For Structuring Human Knowledge." *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, 2008. doi:10.1145/1376616.1376746.

[3] Bordes, Antoine, Xavier Glorot, Jason Weston, and Yoshua Bengio. "A Semantic Matching Energy Function for Learning with Multi-relational Data." *Machine Learning* 94, no. 2 (2013): 233-59. doi:10.1007/s10994-013-5363-6.

[4] Hoffart, Johannes, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. "YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia." *Artificial Intelligence* 194 (2013): 28-61. doi:10.1016/j.artint.2012.06.001.

[5] Newman, M. E. J. "Power Laws, Pareto Distributions and Zipfs Law." *Contemporary Physics* 46, no. 5 (2005): 323-51. doi:10.1080/00107510500052444.

[6] Vosoughi, Soroush, Deb Roy, and Sinan Aral. "The Spread of True and False News Online." Science 359, no. 6380 (2018): 1146-151. doi:10.1126/science.aap9559.

[7] Angeli, G. and C. Manning. "Philosophers are Mortal: Inferring the Truth of Unseen Facts." *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, (Aug. 2013): 133-142. Sofia, Bulgaria: Association for Computational Linguistics.

[8] Bordes, Antoine, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. "Translating Embeddings for Modeling Multi-relational Data." *Advances in Neural Information Processing Systems* 26 (2013): 2787-2795.

[9] Nadeau, David, and Satoshi Sekine. "A Survey of Named Entity Recognition and Classification." *Lingvistic Investigationes* 30, no. 1 (2007): 3-26. doi:10.1075/li.30.1.03nad.

[10] Roelleke, Thomas, and Jun Wang. "TF-IDF uncovered: a study of theories and probabilities." *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2008, 435-42. doi:10.1145/1390334.1390409.

[11] Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. "LIBLINEAR: A Library for Large Linear Classification." Journal of Machine Learning Research 9 (2008): 1871-874.

[12] Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space." *Computing Research Repository* 1301.3781 (2013). http://arxiv.org/abs/1301.3781