Mitigating the Effect of Out-of-Vocabulary Entity Pairs in Matrix Factorization for KB Inference

Prachi Jain\textsuperscript{1}, Shikhar Murty\textsuperscript{1}, Mausam\textsuperscript{1} and Soumen Chakrabarti\textsuperscript{2}
\textsuperscript{1} Indian Institute of Technology Delhi
\textsuperscript{2} Indian Institute of Technology Bombay
\{p6.jain, shikhar.murty\}@gmail.com, mausam@cse.iitd.ac.in, soumen.chakrabarti@gmail.com

Abstract
This paper analyzes the varied performance of Matrix Factorization (MF) on the related tasks of relation extraction and knowledge-base completion, which have been unified recently into a single framework of knowledge-base inference (KBI) \cite{Toutanova et al., 2015}. We first propose a new evaluation protocol that makes comparisons between MF and Tensor Factorization (TF) models fair. We find that this results in a steep drop in MF performance. Our analysis attributes this to the high out-of-vocabulary (OOV) rate of entity pairs in test folds of commonly-used datasets. To alleviate this issue, we propose three extensions to MF. Our best model is a TF-augmented MF model. This hybrid model is robust and obtains strong results across various KBI datasets.

1 Introduction
Inference over knowledge bases (KBs) has received significant attention in the last decade. Most early works focus on adapting probabilistic formalisms such as Markov Logic Networks and Bayesian Logic Programs for inferring new KB facts \cite{Schoenmackers et al., 2008; Niu et al., 2012; Raghavan et al., 2012}. These require a set of inference rules, which can be harvested automatically using statistical regularities in KBs \cite{Schoenmackers et al., 2010; Berant et al., 2011; Nakashole et al., 2012; Jain and Mausam, 2016}.

Recent research has integrated both rule learning and fact inference into a joint deep learning framework. This eschews explicit representation and learning of inference rules, and instead employs a way to score a (possibly new) KB fact \((e_1, r, e_2)\) directly. Various algorithms differ in their scoring functions, by using different model assumptions.

This line of research can be further subdivided into two broad categories: matrix factorization and tensor factorization. In both cases the models learn one or more embeddings of the relation \(r\), however, they differ in their treatment of entities \(e_1\) and \(e_2\). TF approaches (e.g., E \cite{Riedel et al., 2013}, TransE \cite{Bordes et al., 2013}, DistMult \cite{Yang et al., 2015}, ComplEx \cite{Trouillon et al., 2016}) learn separate embeddings for \(e_1\) and \(e_2\), whereas MF methods (e.g., F \cite{Riedel et al., 2013} and extensions \cite{McCallum et al., 2017b}) learn an embedding per entity-pair \((e_1, e_2)\).

MF is considered as the leading technique for Relation Extraction (RE), problems where textual relations are available; whereas TF is the model of choice for Knowledge-Base Completion (KBC). Recent work by Toutanova et al. \cite{Toutanova et al., 2015} unified the two tasks of RE and KBC into a single framework, which we call Knowledge-Base Inference (KBI). This makes MF applicable to all KBI datasets. They applied MF to one dataset for KBC (FB15K-237), where it could not obtain results competitive with the state of the art.

Contributions: The main goal of this paper is to study MF’s effectiveness for the general task of KBI and answer three key questions.

1) Does MF perform well for KBI? Our extensive evaluation reveals that MF has an unusually varied performance across various KBI datasets, achieving MRR scores as high as 74% but also as low as zero.

2) What makes MF performance so sensitive? We find that MF’s performance can be explained by dataset sparsity, in the form of the fraction of entity-pairs that are outside the training vocabulary (OOV).

3) Can we improve MF performance to obtain respectable scores in spite of high OOV rates? The original F model has a rather ad hoc way of handling OOVs — each OOV entity pair gets a random embedding. We propose three enhancements.

Our first model learns a single OOV vector. This ignores signals from the constituent entities, as it uses the same vector for all OOV entity pairs. Our second model generates entity pair vectors from the constituent entity embeddings, on the fly, using a generator layer that is trained to output “MF-like” embeddings. While the second model has better performance on OOV test examples, it degrades drastically on test facts where the gold entity pair is seen — rendering it far inferior to a TF model like ComplEx. In response, we propose a hybrid TF-MF model (inspired by \cite{Singh et al., 2015}). This model is robust and obtains strong results across all datasets.

We recognize that number of parameters in an MF model is much larger compared to TF models. When a joint model is trained naively (as proposed in \cite{Singh et al., 2015}), this imbalance causes TF models to not train properly. Our final

\footnotesize\textsuperscript{1}First two authors contributed equally to the paper
hybrid model allows a more controlled flow of information across the constituent models, achieving much better results.

Additionally, we recognize that special care is needed to handle OOV entity-pairs when evaluating MF against TF. Otherwise an MF algorithm may erroneously appear to perform better than it really does, as in the case of F’s performance on FB15K-237 [Toutanova et al., 2015]. We describe the first unified KBI evaluation protocol that can meaningfully compare MF and TF approaches for KBI.

We contribute open-source implementations† of all models and testing protocols for further research.

Table 1: Scoring functions for MF (row 1) and TF models (row 2 onwards). Larger value implies more confidence in the validity of the triple. · denotes dot product, * denotes element-wise multiplication and  denotes the complex conjugate. Re refers to the real part of the complex valued score returned by the ComplEx model.

2 Background and Experimental Setup

Toutanova et al. [2015] proposed a framework that unifies several closely related tasks: knowledge base completion (KBC), link prediction, and relation extraction (RE). We call this the task of Knowledge Base Inference (KBI). In this section, we describe key data sets, existing methods, loss functions, and evaluation protocol and its limitations.

2.1 Datasets

Most KB inference systems have used one or more of four popular KBs for evaluation. These include WN18 (eighteen Wordnet relations [Bordes et al., 2013]) and three datasets over Freebase (FB) – (1) FB15K [Bordes et al., 2013] that has 1,345 relations; (2) FB15K-237, which is a subset of FB15K comprising 237 relations that seldom overlap in terms of entity pairs [Toutanova et al., 2015]; (3) NYT+FB, which, along with FB triples, also includes dependency path-based textual relations from New York Times between entity mentions that are aligned with entries in Freebase [Riedel et al., 2013].

Our literature search reveals that no model has been tested on all datasets. To the best of our knowledge, no paper reports results of E & F models on WN18 or FB15K, TransE on FB15K-237 or NYT+FB, and DM & ComplEx on NYT+FB. To better understand the strengths and weaknesses of each model, we compare all models on all datasets. We also release their open source implementations for further research.

2.2 Previous Approaches

We are given a KB with a set of entities and relations which may include canonical and ("open") textual relations. The KB also contains a set of valid tuples . A tuple consists of a subject entity , object entity , and relation . We use a shorthand to refer to entity pair . The goal in KBI is to predict the validity of a tuple not known to be in the KB. Previous approaches fit continuous representations (loosely, “embeddings”) to entities and relations, so that the belief in the veracity of (subject, relation, object) can be estimated as an algebraic expression (called a scoring function ) involving those embeddings. The scoring functions for the models considered in this work are outlined in Table 1.

Our choice of these models is guided by the fact that these algorithms either form the basis of several recent papers on KB inference or are popular baselines for comparison studies [Toutanova et al., 2015, Trouillon et al., 2016].

Demeester et al. [2016], Rocktäschel et al. [2015], Verga et al. [2017b], Verga et al. [2016], Singh et al. [2015], Nguyen et al. [2016], Xie et al. [2017]). Some models learn matrix embeddings instead of vectors [Nickel et al., 2011; Socher et al., 2013]. We don’t study these, as they are typically outperformed by the models implemented in this paper.

Note that some approaches (called tensor factorization or TF) embed each entity individually, whereas others (called matrix factorization or MF) embed pairs of entities. F is an MF model, since it uses the embeddings, while the others are TF models.

Loss functions: The models are trained such that tuples observed in the KB have higher scores than unobserved ones. Several loss functions have been proposed; we implement three common ones in this work: log-likelihood based loss [Toutanova et al., 2015], max margin loss [Bordes et al., 2013], and negative loglikelihood of the logistic model [Trouillon et al., 2016]. All three loss functions sample a negative set for every tuple, computed as , i.e., tuples formed by uniformly sampling entities that are not a priori known to be valid. Similarly, the set is sampled.

Note that since MF models operate over entity pairs, they do not need two sets. They use one set where new entity pairs are sampled such that . These negative entity pairs are sampled only from the entity pairs found in , since embeddings for only those pairs get learned. We tried all the losses with all models on all datasets and report the best scores in the paper.

2.3 Standard Evaluation Protocol

To run the experiments at scale, we follow the most common automatic evaluation protocol for KBIsystems. In this method the KB is split into training ( ) and test tuples ( ). The system can access only during training. For each test tuple, a query is issued to the trained model . The model then ranks all entities not in the KB by decreasing involving those embeddings. A higher rank of denotes a better performance of the model. Common metrics used to compare algorithms are mean reciprocal rank (MRR) and the percentage of obtained in top 1 (HITS@1) and 10 (HITS@10) results.

The testing procedure is typically run with two modifications. First, it is possible that the test fold is not complete: some of the known tuples yield known tuples . It is unfair to penalize the model for predicting
Table 2: The first five rows compare 5 models on 4 datasets using the standard evaluation protocol. The 6th row shows F’s performance using our proposed KBI evaluation protocol. The last 2 rows report results of 2 most-frequent sanity-check baselines.

<table>
<thead>
<tr>
<th>Model</th>
<th>FB15K MRR</th>
<th>FB15K-237 MRR</th>
<th>WN18 MRR</th>
<th>NYT+FB MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hits@1</td>
<td>Hits@10</td>
<td>Hits@1</td>
<td>Hits@10</td>
</tr>
<tr>
<td>1 DistMult</td>
<td>60.82</td>
<td>46.51</td>
<td>84.78</td>
<td>37.21</td>
</tr>
<tr>
<td>2 DistMult</td>
<td>22.86</td>
<td>16.4</td>
<td>35.04</td>
<td>3.57</td>
</tr>
<tr>
<td>3 E</td>
<td>43.11</td>
<td>24.99</td>
<td>71.97</td>
<td>28.62</td>
</tr>
<tr>
<td>4 TransE</td>
<td>33.62</td>
<td>22.07</td>
<td>60.20</td>
<td>28.01</td>
</tr>
<tr>
<td>5 F (Old eval)</td>
<td>13.35</td>
<td>9.45</td>
<td>17.05</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 3: Comparison Under Standard and Sanitized KBI Evaluation Protocol

In this section, we first present a detailed study of previous approaches under standard evaluation protocols, which exposes the limitations of existing evaluation protocols. We then propose a sanitized protocol which evaluates KBI models more accurately. We further add a few baselines overlooked in prior work, and present results adjusted for KBI evaluation.

Implementation details: We implement all algorithms in a common framework written using Keras/Theano [Chollet, 2015]. We use embeddings in $\mathbb{R}^{100}$ throughout, trained using Adagrad. 200 random negative samples are drawn per positive tuple. Margin $\gamma$ is 1 for max margin methods. Entity and entity-pair vectors are re-normalized to unit norm after each batch update [Yang et al., 2015]. Batch size is 20,000. Models are trained up to 200 epochs, but with early stopping on a validation set to prevent from overfitting. We train each model on each dataset using log-likelihood (LL), max-margin (MM) and logistic (L) loss functions and pick the best loss function on each dataset using log-likelihood (LL), max-margin (MM) validation set to prevent from overfitting. We train each model

Model performance with the Standard protocol: The first five rows of Table 2 report standard protocol performance of all the models across the datasets. E has good performance on FB15K-237, whereas TransE gets good scores on FB15K, however ComplEx emerges the most robust. For TF models on three datasets (FB15K, FB15K-237, WN18) our experiments are able to replicate (or improve upon) most reported results [Yang et al., 2015; Bordes et al., 2013; Toutanova et al., 2015]. Since NYT+FB uses a new test fold, and F hasn’t been tested on other datasets, those results cannot be directly compared against previous work.

We find that F outperforms ComplEx on NYT-FB dataset by wide margins and does not perform as well as ComplEx on the rest. It appears that a qualitative analysis of ComplEx vs F will shed light on their relative strengths and weaknesses. Our analysis reveals a limitation in the standard evaluation protocol that can inflate F’s performance scores for OOV entity pairs.

The problem with the Standard protocol: Recall the second modification from standard protocol. When ranking possible entity pairs $e_2$ using the score $\phi(e_1^+, r^*, e_2)$ from MF models, the standard protocol considers a subset $E_2$ of all entities in $\mathcal{E}$. This is because many entity pair embeddings $(e_1^+, e_2)$ are not even trained in the model, and hence their scores will be meaningless. We call these OOV entity pairs. $E_2$ contains all entities for which the entity pair $(e_1^+, e_2)$ is trained. Additionally, all such $e_2$s that are gold entities for some test query $(e_1^+, r^*, e)$ are also added to $E_2$. If these are not trained, a random vector is assumed for them.

Proposed Sanitized protocol: A correct KBI evaluation protocol must assume all OOV entities at the same rank, and output the average value over all possible rankings for them. In our sanitized protocol, we assume one random OOV entity pair $(e_1^+, e_{ov})$, identify all $e_2 \in \mathcal{E}$ that are OOV, assign them all the same score from the model and compute aggregate...
However, it continues to have the best numbers (among all trained models) for NYT+FB, under the sanitized protocol. MFreq(e_2|r^*) beats F model significantly suggesting that while F is the best model on that dataset, it is not good enough. We explore this further in the next section.

<table>
<thead>
<tr>
<th>Dataset</th>
<th></th>
<th></th>
<th>ep OOV (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB15K</td>
<td>14,951</td>
<td>1,345</td>
<td>68.70</td>
</tr>
<tr>
<td>FB15K-237</td>
<td>14,541</td>
<td>237</td>
<td>100.00</td>
</tr>
<tr>
<td>WN18</td>
<td>40,943</td>
<td>18</td>
<td>99.52</td>
</tr>
<tr>
<td>NYT+FB</td>
<td>24,528</td>
<td>4,111</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Table 4: No. of distinct entities, no. of relations and entity pair OOV rate, i.e., percentage of tuples in test set, whose entity pairs (ep) weren’t seen while training.

4 Toward a Robust MF Model

The previous section highlights the importance of OOV entity-pairs in the performance of MF models. We now present a series of models to gracefully handle entity pair OOVs within MF.

4.1 MF with a Trained OOV Vector (F_{vecOOV})

A natural extension to F is to explicitly model an OOV entity-pair vector. In particular, we represent a vector e_{OOV} for F and e_{OOV} for TF. This modification means that all facts with an OOV entity-pair will have the same score. OOV vectors can be trained in many ways. We develop two baselines that don’t train the vectors explicitly. The first

\[ \{t|t = (e_1, r^*, e_2) \land t \in T_{ts} \}. \]

A similar baseline orders each entity e_2 based on its frequency of occurrence with e_1^*, i.e., based on cardinality of the set \( \{t|t = (e_1^*, r, e_2) \land t \in T_{ts} \} \). We name these baselines MFreq(e_2|r^*) and MFreq(e_2|e_1^*) respectively.

The last two rows of Table 2 report the performance of these baselines. It is satisfying to see that for FB15K and WN18 datasets, ComplEx outperforms the baselines by large margins. However, for FB15K-237, ComplEx is only marginally better than MFreq(e_2|r^*). A closer analysis reveals that this dataset is constructed so that there is minimal entity-pair overlap between relations. How would any model, then, predict the best e_2 for a query \((e_1^*, r^*, ?)\)? If entity pairs have not been repeated much, a natural approach may just find the most frequent entities seen with the relation and order based on frequency. We checked some high MRR predictions made by ComplEx and found that often questions, like, what is the language of a specific website, were answered correctly as English. This is likely not because ComplEx figured out the language of each website, but because English was the most frequent one in the dataset.

We also observe that E’s performance remains broadly similar to the performance of MFreq(e_2|r^*). We attribute this to E’s scoring function, since given e_1^* and r^*, the only term relevant for ranking e_2 is e_2^* \cdot r_{oo}, i.e., the model looks for compatibility with r^* and ignores e_1^* completely.

Finally, for NYT+FB, under the sanitized protocol, MFreq(e_2|e_1^*) beats F model significantly suggesting that while F is the best model on that dataset, it is not good enough. We explore this further in the next section.
Table 5: Results on F model after explicitly modeling OOV vectors. OOV training outperforms other baselines, especially for NYT+FB. Results on FB15k-237 not reported, due to 100% OOV rate.

<table>
<thead>
<tr>
<th>Model</th>
<th>FB15k MRR</th>
<th>Hits@10</th>
<th>WNT8 MRR</th>
<th>Hits@10</th>
<th>NYT+FB MRR</th>
<th>Hits@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>F (random)</td>
<td>13.35</td>
<td>17.03</td>
<td>0.14</td>
<td>0.20</td>
<td>74.34</td>
<td>80.01</td>
</tr>
<tr>
<td>F (average)</td>
<td>18.27</td>
<td>24.62</td>
<td>0.13</td>
<td>0.16</td>
<td>71.65</td>
<td>76.80</td>
</tr>
<tr>
<td>F (trained)</td>
<td>20.21</td>
<td>27.42</td>
<td>0.27</td>
<td>0.38</td>
<td>81.51</td>
<td>93.67</td>
</tr>
<tr>
<td>F (generated)</td>
<td>13.51</td>
<td>22.67</td>
<td>0.80</td>
<td>1.22</td>
<td>0.20</td>
<td>0.10</td>
</tr>
</tbody>
</table>

Table 6: Performance segregated by OOV and non-OOV test queries on FB15k. F+C (RAL) performs best on both OOV and non-OOV, baseline (Table 5 line 1) assigns a random value to $\vec{e}_{p_{ooov}}$. The second one called the average baseline computes $\vec{e}_{p_{ooov}}$ as the average of all $(e_1, e_2)$ pairs that occur only once in training (Table 5 line 2).

We also propose a procedure to train $\vec{e}_{p_{ooov}}$ (Table 5 line 3). The broad motivation is to score a known tuple higher than a tuple with an OOV. To ensure this, we add $\vec{e}_{p_{ooov}}$ in the $N_{eq}$ set for each train tuple. This encourages the model to learn embeddings such that $\phi^F(r, \vec{p}_{12}) > \phi^F(r, \vec{e}_{p_{ooov}})$. Thus, we ensure that the performance of $F$ is maintained when a gold test entity pair is seen during training. Table 3(b) illustrates an example where the correct answer (New York) is seen with Tina Fey and OOV training doesn’t displace its position.

To assess the effectiveness of the $F_{vecOOV}$ model, we breakdown its performance on subset of test queries that have OOVs and non-OOV gold entity pairs. This analysis is meaningful only for FB15k, since other datasets have extreme entity-pair OOV rate (see Table 4). Clearly, as Table 6 shows, while $F_{vecOOV}$ has extremely poor performance on OOVs (and thus weak performance overall), it performs decently on non-OOVs. We attribute this to the fact that the $F_{vecOOV}$ is designed with the inductive bias that facts with OOV embeddings are worse than facts with seen entity-pair embeddings. Another shortcoming of this model is that it ignores all information present in constituent entities of an OOV entity pair.

4.2 Generate OOV Entity Pair Vectors ($F_{genEP}$)

To fix the above shortcomings, we propose an enhancement to aid $F$ in performing well on facts with OOV entity pairs. The main insight is to generate an informed OOV entity pair embedding by leveraging the information in the constituent entities. We would like these generated entity pair embeddings to be similar to what $MF$ would have produced had these entity pairs been observed. To this end, we train a generator layer that is applied on the concatenation of an entity pair’s constituent entities’ vectors to produce entity pair embeddings.

We obtain such a generator layer as a by product of training a model with loss function $\mathcal{L}(\vec{e}_{p_{12}}, \vec{r}, e_1, e_2) = \mathcal{L}^F(\vec{e}_{p_{12}}, \vec{r}) + \mathcal{L}^C(\vec{e}_{p_{12}}, e_1, e_2)$. Here $\mathcal{L}^F$ is the loss of the vanilla $F$ model and $\mathcal{L}^C = \|\vec{e}_{p_{12}} - \vec{p}_{12}\|^2$, is regression loss, which we use to guide the generator layer to generate $F$ like embeddings. We define $\vec{e}_{p_{12}} = f(P([e_1, e_2]))$, where the generator layer $P \in \mathbb{R}^{d_x \times d}$ and $f$ can be any activation function.

At test time, we concatenate the trained embeddings of constituent entities and use the learnt generator layer to obtain an embedding for an entity pair. Note that use generated entity pair embedding only for OOV entity-pairs, and use the trained embeddings of seen entity pairs from MF model.

At train time, we initialize the model with trained MF entity pair embeddings, MF relation embeddings and DistMult entity embeddings. We do not use any activation function ($f$) as it led to a slight decrease in performance. Table 6 shows that $F_{genOOV}$ indeed has improved performance on OOV test facts. However, the performance on seen test facts drops significantly, leading to a weak overall performance.

4.3 Using TF to Guide MF

We now extend MF (from Section 4.1) based on the following two observations — (1) TF models like ComplEx perform robustly on both OOV and non-OOV test facts (see Table 6). (2) Singh et al. [2015] show that MF and TF models have complimentary strengths. Could we use a TF model (such as ComplEx) to guide MF to perform better on OOV test facts? Background on TF augmented MF models: Recall that Singh et al. [2015] find that the two models have complimentary strengths. In response, they developed a TF augmented MF model which outperforms all other models on artificial datasets and NYT+FB. Their best model (F+E) uses the scoring function $\phi^{E+F} = \sigma(\phi^E + \phi^F)$, where $\sigma$ is the sigmoid function. We call this model an additive score (AS) model, since the scores ($\phi$) of two models are added. Early works of Reidel et al. [2013] also experiment with a similar model for NYT+FB. Later, Toutanova et al. [2015] implement an AS F+E+DM model and tested it on FB15K-237.

We are motivated to develop an extension to MF that leverages TF to perform gracefully on both OOV and non-OOV test facts. For brevity, we refer to these MF extensions as joint MF-TF models, since they involve a TF model to guide the training of MF. Does an Additive Score model meet this requirement?

Additive loss (AL) joint model: Preliminary investigations (Table 6) reveal that additive score models can suffer substantial loss in performance for both OOV and non-OOV test facts. Table 8 shows drop in performance in the ComplEx component when trained jointly in additive score F+ComplEx model. It clearly shows that ComplEx’s performance can reduce drastically due to joint training. A primary reason is that F scores overshadow ComplEx scores, since the scoring function in F involves a product of 2 small numbers and ComplEx involves a product of 3 small numbers. Moreover, the

4To calibrate them, we tried standardizing scores obtained from pre-trained models. We also tried to learn a linear function to push ComplEx and F model scores to the same range simultaneously. We also tried sharing of relation parameters to allow information to flow from ComplEx to MF. Unfortunately, none of the approaches were robust across datasets.
number of parameters in MF models (vectors for entity pairs) significantly outnumber those in TF models (vectors for entities). This can lead to significant overfitting.

In response, we develop a different class of joint models in which instead of adding the scores ($\phi$s), we add their loss functions: $L^{MF} + L^{TF}$. We name these additive loss joint models (AL). We expect this to be more resilient to overfitting, since the joint loss expects each model’s individual loss to decrease as much as possible. One may note that AL style of training is equivalent to training the models separately. However, joint training makes other extensions possible, such as regularization.

**Regularized additive loss (RAL):** We extend the vanilla AL joint model to a regularized joint model in which the parameters of MF model are L2-regularized. We expect this regularization to encourage a reduction in overfitting caused due to the large number of MF parameters. Overall, our final joint model has the loss function:

$$L^{MF} + L^{TF} = L^{MF}(\theta^{MF}) + L^{TF}(\theta^{TF}) + \lambda \left\| \theta^{MF} \right\|_2$$

At test time, for a query ($e_1^*, r^*, ?$) an AL model cannot simply add the scores, since some entity-pairs may be OOVs. We develop various backoff cases, reminiscent of traditional backoff in language models [Manning and Schütze, 2001]. For every $e_2$:

- **Case 1:** ($e_1^*, e_2^*) \in T_{tr}$. Score of tuple is $\phi^{TF}(e_1^*, r^*, e_2^*) + \phi^{MF}(e_1^*, r^*, e_2^*)$.
- **Case 2:** ($e_1^*, e_2$) $\notin T_{tr}$, but $e_2$ is seen in training. Score of tuple is $\phi^{TF}(e_1^*, r^*, e_2^*) + \phi^{MF}(e_oov, r^*, e_2^*)$.
- **Case 3:** $e_2$ is not seen in training. Score of tuple is $\phi^{TF}(e_1^*, r^*, e_oov) + \phi^{MF}(e_oov, r^*, e_oov)$.

**Results:** Table 6 shows that RAL F+ComplEx performs well on OOV and non-OOV test queries. Regularization penalty $\lambda$ is tuned over a small devset from within the training set.

**Robustness:** RAL F+ComplEx performs well for OOV & Non-OOV test cases, hence is robust to all testing conditions. To validate the hypothesis we evaluate the model on all the datasets in background section. Table 7 compares its performance with individual models as well as other joint models.

We find that different additive score models (rows 3–5) perform well on some datasets, but are not robust across them. For example, in FB15K-237 none of these are able to match up to ComplEx’s performance. We attribute this to overfitting by F, which makes the model believe that $\phi^F$ is predicting the tuple very well. This lets F override TF and reduces the joint model’s need to learn the best TF model(s). Note that row 3 and row 5 are the models reported in (Singh et al. [2015]) and [Toutanova et al., 2015], respectively.

Rows 6 and 7 report the results of additive loss F+ComplEx models, both without and with regularization. As anticipated, adding the losses improves performance since both models get trained well. Moreover, regularization also helps considerably since now the model is not overwhelmed by too many F parameters. RAL version of F+ComplEx achieves the best scores in all datasets, and is the state-of-the-art among all reported numbers for FB15k-237.

Row 8 of Table 7 also shows the accuracy of an oracle model that, for every test query, post-facto selects the model with the more accurate score (between ComplEx and F). This upper bounds the performance expected from a perfect joint ComplEx+F model, fixing the constituents. We find that the oracle is only 4-5 MRR percentage points better than our best model for two datasets, and the differences are much less for the other two. Overall, it suggests that our proposed joint model obtains a strong robust performance.

5 Other Related Work

The original F model has been extended to incorporate first order logic rules (Rocktaschel et al. [2015]; Deenest et al. [2016]), to predict for relations not seen at training time [Verga et al., 2016; McCallum et al., 2017a], etc. It has also been extended to generate embedding of a new entity-pair on the fly (Verga et al. [2017b]). However their work is different from ours, since, at test time, they expect knowledge of several tuples between the same entity pair. Similarly, other TF models also exist, for example, ParaFac [Harshman, 1970], Rascal (Nickel et al. [2011]) and NTN [Socher et al., 2013]. These are older models which have been outperformed by models evaluated in this paper. More recent models have also been introduced such as a model using holographic embeddings (Nickel et al. [2016]). ComplEx model is shown to be equivalent to the holographic models [Wang et al., 2017]. The joint model formulation can incorporate new models sequenc-

Table 7: Performance of joint models. AL = additive loss. AS = additive score. F+ComplEx combined with regularized additive loss (RAL) is the highest scorer as well as most robust across all datasets.
lessly and improve even further. The learned embeddings can use additional information such as typing [Chang et al., 2014; Jain et al., 2018], can be used to mine logical rules [Yang et al., 2015] and induce schemas [Nimishakavi et al., 2016].

6 Discussion and Future Work

We now list two observations that suggest important directions for future research in KB inference.

Dataset Characteristics: Our work subjects datasets to natural sanity checks. First, we introduce two most frequent baselines (Table 2) to understand the nature of the KBs. Second, we compute entity-pair OOV rates (Table 4) as a rough predictor of the relative success of the TF and MF families. Finally, in Table 9, we report the singleton and doubleton percentages (for entity pairs). A singleton is an entity-pair occurring only once in the data ($T_s \cup T_r$) and a doubleton is an entity pair that occurs exactly twice. Doubletons have a strong effect in the scenario painted in Table 3.

We find that most datasets have an idiocyncrasy, which raises the question whether they are good representatives for naturally occurring KBs. In particular, WN18 and FB15K-237 have near 100% entity-pair OOV rates, unlike to be the case in real KBs. In FB15K-237 the best models are not much better than MFreq($e_2$ | $r^+$) baseline. This is because the dataset is artificially constructed to avoid relations with entity-pair overlap. But, this reduces its ability to make many interesting inferences. For NYT+FB, MFreq($e_2$ | $e_1$) performance has a strong performance with 95% score on HITS@10. Moreover, learned models are able to improve its MRR by only about four percentage points. Statistics in Table 9 reveal that this could be because the dataset has an unusually high number of entity-pair doubletons: it is the only data set where doubletons by far outnumber singletons. It is unlikely that such a distribution occurs in a naturally occurring dataset. FB15K appears to pass our sanity tests. We believe that focus on better datasets will likely help us in better progress on KB inference.

Path based inference: In KBs, a common type of inference is based on relation paths (or Horn-clauses), e.g., (Michael Jordan, teaches at, Berkeley) and (Berkeley, is located in, California) implies (Michael Jordan, teaches in, California). To assess the ability of inference models to automatically learn such relation paths, we tested them on artificial datasets, where we provided many instances of two-hop paths with relations $r_1$ and $r_2$ implying a third relation $r_3$. We find that none of the four models are effective at predicting such relations. A study similar to ours comparing the latest models that train over relation paths [Gua et al., 2015; García-Durán et al., 2015; Toutanova et al., 2016] will benefit our understanding of path-based inference.

### Table 9: The fraction of entity-pairs occurring exactly once and exactly twice. NYT+FB has an unusual distribution.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Singleton Rate</th>
<th>Doubleton Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB15K</td>
<td>83.83%</td>
<td>12.19%</td>
</tr>
<tr>
<td>FB15K-237</td>
<td>90.52%</td>
<td>8.64%</td>
</tr>
<tr>
<td>WN18</td>
<td>99.80%</td>
<td>0.20%</td>
</tr>
<tr>
<td>NYT+FB</td>
<td>8.06%</td>
<td>59.04%</td>
</tr>
</tbody>
</table>

7 Conclusion

We present the first study on the effectiveness of MF for KBI, which unifies Relation Extraction and Knowledge Base Completion. After replacing the standard evaluation protocol with our sanitized proposal, we find that MF’s performance is highly varied — it obtains MRR scores from 0 to 74% on different datasets. We also propose two simple frequency baselines, and are surprised to find that MF’s performance is worse than than the better baseline in all domains! Further analysis reveals that MF performs poorly at high-OOV rates. We develop a series of extensions aimed at mitigating the effect of OOVs in MF. MF’s performance improves by training OOV embeddings. Our most successful model uses ComplEx to augment our improved version of MF via a regularized additive loss. This hybrid model is highly robust and has the best performance on all datasets. Note that a basic MF added to TF isn’t robust – only an “OOV trained” MF, when integrated with TF, attains good performance.

We release our code for all models and evaluation protocols for further use by research community.²

Acknowledgements

This work is supported by Google language understanding and knowledge discovery focused research grants, a Bloomberg award, an IBM SUR award, and a Visvesvaraya faculty award by Govt. of India to the third author. It is also supported by a TCS Fellowship to the first author. We thank Microsoft Azure sponsorships, and IIT Delhi HPC facility for computational resources. We also acknowledge the insightful comments by Sameer Singh, Dan Weld and Oren Etzioni on an early draft of the work.

References


²https://github.com/dair-iitd/KBI


