JEL: Applying End-to-End Neural Entity Linking in JPMorgan Chase

Wanying Ding, Vinay K. Chaudhri, Naren Chittar, Krishna Konakanchi
JPMorgan Chase & Co.
{wanying.ding, vinay.chaudhri, naren.chittar}@jpmchase.com, krishna.k.konakanchi@chase.com

Abstract

Knowledge Graphs have emerged as a compelling abstraction for capturing key relationship among the entities of interest to enterprises and for integrating data from heterogeneous sources. JPMorgan Chase (JPMC) is leading this trend by leveraging knowledge graphs across the organization for multiple mission critical applications such as risk assessment, fraud detection, investment advice, etc. A core problem in leveraging a knowledge graph is to link mentions (e.g., company names) that are encountered in textual sources to entities in the knowledge graph. Although several techniques exist for entity linking, they are tuned for entities that exist in Wikipedia, and fail to generalize for the entities that are of interest to an enterprise. In this paper, we propose a novel end-to-end neural entity linking model (JEL) that uses minimal context information and a margin loss to generate entity embeddings, and a Wide & Deep Learning model to match character and semantic information respectively. We show that JEL achieves the state-of-the-art performance to link mentions of company names in financial news with entities in our knowledge graph. We report on our efforts to deploy this model in the company-wide system to generate alerts in response to financial news. The methodology used for JEL is directly applicable and usable by other enterprises who need entity linking solutions for data that are unique to their respective situations.

Introduction

Knowledge Graphs are being used for a wide range of applications from space, journalism, biomedicine to entertainment, network security, and pharmaceuticals. Within JP Morgan Chase (JPMC), we are leveraging knowledge graphs for financial applications such as risk management, supply chain analysis, strategy implementation, fraud detection, investment advice, etc. While leveraging a knowledge graph, *Entity Linking* (EL) is a central task for semantic text understanding and information extraction. As defined in many studies (Zhang et al. 2010; Eshel et al. 2017; Kolitsas, Ganea, and Hofmann 2018), in an EL task we link a potentially ambiguous *Mention* (such as a company name) with its corresponding *Entity* in a knowledge graph. EL can facilitate several knowledge graph applications, for example, the mentions of company names in the news are inherently ambiguous, and by relating such mentions with an internal knowledge graph, we can generate valuable alerts for financial analysts. In Figure 1, we show a concrete example in which the name "Lumier" has been mentioned in two different news items. "Lumier"s are two different companies in the real world, and their positive financial activities should be brought to the attention of different stakeholders. With a successful EL engine, these two mentions of "Lumier"s can be distinguished and linked to their corresponding entities in a knowledge graph.

Prior work on EL has been driven by a number of standard datasets, such as CoNLYAGO (Suchanek, Kasneci, and Weikum 2007), TAC KBP[1], DBpedia[2], and ACE[3]. These datasets are based on Wikipedia, and are therefore, naturally coherent, well-structured and rich in context (Eshel et al. 2017). We face the following problems when we use these methods for entity linking for our internal knowledge graph:

1) Wikipedia does not cover all the entities of financial interest. For example, as of this writing, the startup "Lumier" mentioned in Figure 1 is not present in Wikipedia, but it is of high financial interest as it has raised critical investment from famous investors.
2) Lack of context information. Many pre-trained models achieve great performance by leveraging rich context data from Wikipedia (Ganea and Hofmann 2017). For JPMC internal data, we do not have information comparable to Wikipedia to support re-training or fine-tuning of existing models.

To address the problems identified above, we built a novel entity linking system, JEL, to link mentions of company names in text to entities in our own knowledge graph. Our model makes the following advancements on the current state-of-the-art:

1) We do not rely on Wikipedia to generate entity embeddings. With minimum context information, we compute entity embeddings by training a *Margin Loss* function.
2) We deploy the *Wide & Deep Learning* (Cheng et al. 2016) to match character and semantic information respectively.

[1]https://www.ldc.upenn.edu/collaborations/current-projects/tac-kbp
[2]https://wiki.dbpedia.org/develop/datasets
[3]https://catalog.ldc.upenn.edu/LDC2006T06

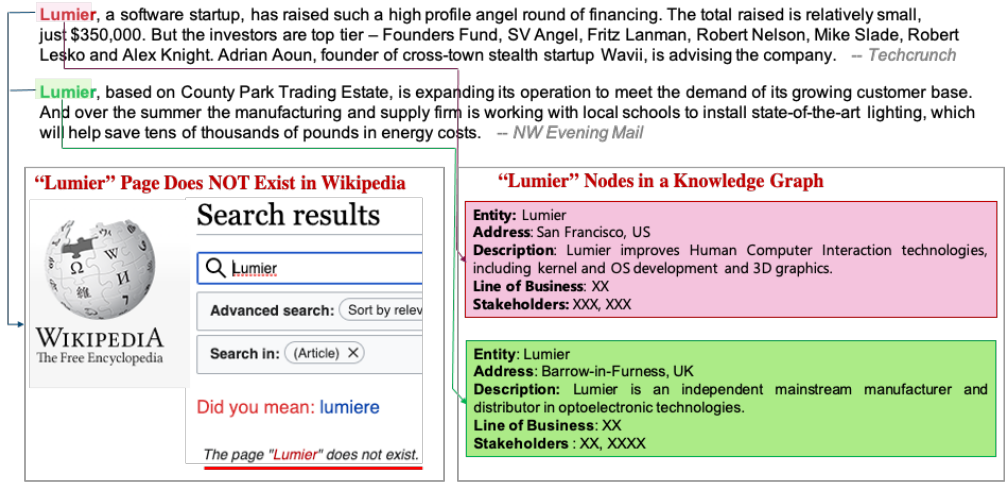

Figure 1: Example for Entity Linking

Unlike other deep learning models (Martins, Marinho, and Martins 2019; Kolitsas, Ganea, and Hofmann 2018; Ganea and Hofmann 2017), JEL applies a simple linear layer to learn character patterns, making the model more efficient both in the training phase and inference phase.

Problem Definition and Related Work

Problem Definition

We assume a knowledge graph (KG) has a set of entities E. We further assume that W is the vocabulary of words in the input documents. An input document D is given as a sequence of words: $D = \{w_1, w_2, ..., w_d\}$ where $w_k \in W, 1 \leq k \leq d$. The output of an EL model is a list of T mention-entity pairs $\{(m_i, e_i)\}_{i \in \{1,T\}}$, where each mention is a word subsequence of D, $m_i = w_l, ..., w_r, 1 \leq l \leq r \leq d$, and each entity $e_i \in E$. The entity linking process involves the following two steps (Ceccarelli et al. 2013).

1) **Recognition**. Recognize a list of mentions m_i as a set of all contiguous sequential words occurring in D that might mention some entity $e_i \in E$. We adopted spaCy[4] for mention recognition.

2) **Linking**. Given a mention m_i, and the set of candidate entities, $C(m_i)$ such that $|C(m_i)| > 1$, from the KG, choose the correct entity, $e_i \in C(m_i)$, to which the mention should be linked. We focus on solving the linking problem in this paper.

Popular Methods

Entity Linking is a classical NLP problem for which the following techniques have been used: *String Matching*, *Context Similarity*, *Machine Learning Classification*, *Learning to Rank*, and *Deep Learning*. In the following several paragraphs, we will briefly discuss each of them.

[4]https://spacy.io/

– **String Matching Methods.** String matching measures the similarity between the mention string and entity name string. We experimented with different string matching methods for name matching, including Jaccard, Levenshtein, Ratcliff-Obershelp, Jaro Winkler, and N-Gram Cosine Simiarity, and found that n-gram cosine similarity achieves the best performance on our internal data. However, pure string-matching methods breakdown when two different entities share similar or the same name (as shown in Figure 1) which motivates the need for better matching techniques.

– **Context Similarity Methods.** Context Similarity methods compare similarities of respective context words for mentions and entities. The context words for a mention are the words surrounding it in the document. The context words for an entity are the words describing it in the KG. Similarity functions, such as Cosine Similarity or Jaccard Similarity, are widely used to compare the two sets of context words (Cucerzan 2007; Mihalcea and Csomai 2007), and then to decide whether a mention and an entity should be linked.

– **Machine Learning Classification.** Many studies adopt machine learning techniques for the EL task. Binary classifiers, such as Naive Bayes (Varma et al. 2009), C4.5 (Milne and Witten 2008), Binary Logistic classifier (Han, Sun, and Zhao 2011), and Support Vector Machines (SVM) (Zhang et al. 2010), can be trained on mention-entity pairs to decide whether they should be linked.

– **Learn to Rank Methods.** As a classification method will generate more than one mention-entity pairs, many systems use a ranking model (Zheng et al. 2010) to select the most likely match. Learning to Rank (LTR) is a class of techniques that supplies supervised machine learning to solve ranking problems.

– **Deep Learning Methods.** Deep learning has achieved success on numerous tasks including EL (Sun et al. 2015;

Huang, Heck, and Ji 2015; Francis-Landau, Durrett, and Klein 2016). One specific model (Kolitsas, Ganea, and Hofmann 2018) uses two levels of Bi-LSTM to embed characters into words, and words into mentions, and calculates the similarity between a mention vector and a pre-trained entity vector (Ganea and Hofmann 2017) to decide whether they match.

Proposed Framework

Entity Embedding

Most public entity embedding models (He et al. 2013; Yamada et al. 2016; Ganea and Hofmann 2017) are designed for Wikipedia pages and require rich entity description information. In our case, each entity has a short description that is insufficient to support a solid statistical estimation of entity embeddings (Mikolov, Yih, and Zweig 2013). To address this limitation, we use a *Triplet Loss* model to generate our own entity embeddings from pre-trained word embedding models with limited context information support.

Entity Embedding Model. To prepare training data for this model, we select 10 words that can be used as positive examples and 10 words that can be used as negative example for each entity. To select the positive examples, we score each entity's description words with tf-idf, and select the words with 10 highest scores. To select the negative examples, we randomly select from words that do not appear in this entity's description. Thus, for each entity, we can construct 10 $<$ entity, positive-word, negative-word $>$ triplets to feed into triplet loss function formulated as Equation 1 below.

$$Loss = \sum_{i=1}^{N} [||f_i^a - f_i^p||_2^2 - ||f_i^a - f_i^n||_2^2 + \alpha]_+ \quad (1)$$

where f_i^a is the vector of an anchor that we learn, f_i^p is the vector from a positive sample, and f_i^n is the vector from a negative sample, α is the margin hyper-parameter to be manually defined. We train the entity embedding vectors (f^a). We use off-the-shelf word embedding vectors(f^p and f^n) from the fastText language model. In our experiments, $\alpha = 2.0$ led to the best performance.

Entity Embedding Validation. To validate the entity embeddings, we choose five seed companies from different industries — "Google DeepMind", "Hulu", "Magellan Health", "PayPal Holdings", "Skybus Airlines". We next select their ten nearest neighbors (as shown in Figure 2). We calculate a t-Distributed Stochastic Neighbor Embedding (t-SNE) to project the embeddings into a 2-dimension space. As clearly shown in Figure 2, five seed companies from different industries are clearly separated in space. For "Google DeepMind", we can find that all its neighbors [5] are, as expected, Artificial Intelligence and Machine Learning companies. This visualization gives us a sanity check for our entity embeddings.

[5] SambaNova Systems, Lumier, ParallelM, CTRL-labs, Coherent AI, etc.

Entity Linking

Two factors affect an EL model's performance: *Characters* and *Semantics*.

- **Characters**: "Lumier" will be easily distinguished from " ParallelM" because they have completely different character patterns. These patterns can be easily captured by a wide and shallow linear model.
- **Semantics**. In Figure 1, "Lumier(Software)" can be distinguished from "Lumier (LED)" because they have different semantic meanings behind the same name. These semantic differences can be captured by a deep learning model.

To combine the two important factors listed above, we develop a Wide&Deep Learning model (Cheng et al. 2016) for our EL task (shown in Figure 4).

Wide Character Learning. Unlike many other approaches (Kolitsas, Ganea, and Hofmann 2018; Lample et al. 2016) that apply character embeddings to incorporate lexical information, we apply a wide but shallow linear layer for the following two reasons. First, embedding aims to capture an item's semantic meanings, but characters naturally have no such semantics. "A" in "Amazon" does not have any relationship with "A" in "Apple". Second, as embedding layer involves more parameters to optimize, it is much slower in training and inference than a simple linear layer.

Feature Engineering. Many mentions of an entity exhibit a complex morphological structure that is hard to account for by simple word-to-word or character-to-character matching. Subwords can improve matching accuracy dramatically. Given a string, we undertake the following processing to maximize morphological information we can get from subwords.

1) Clean a string, convert it to lower case, remove punctuation, standardize suffix, etc. For example, "PayPal Holdings, Inc. " will change to "paypalhlds".
2) Pad the start and end of the string;"paypalhlds" will be converted to "*paypalhlds*".
3) Apply multiple levels of n-gram ($n \in [2, 5]$) segmentation;"*paylpalhlds* " will be { *p, ay,..., lhlds, hlds* }.
4) Append original words, *paypal* and *hlds*, to the token list.

Wide Character Learning. We applied a *Linear Siamese Network* (Bromley et al. 1994) for wide character learning. We implemented two identical linear layers with shared weights (as shown in the left part of Figure 4). With this architecture, similar inputs, T_m and T_e, will generate similar outputs, Y_m and Y_e. We applied the Euclidean distance to estimate output's similarity.

$$D_{syx} = d(\mathbf{Y_m}, \mathbf{Y_e}) = \sqrt{\sum_{i=1}^{n} (Y_{m_i} - Y_{e_i})^2} \quad (2)$$

Deep Semantic Embedding. We embed the mentions into vectors so that we can mathematically measure similarities

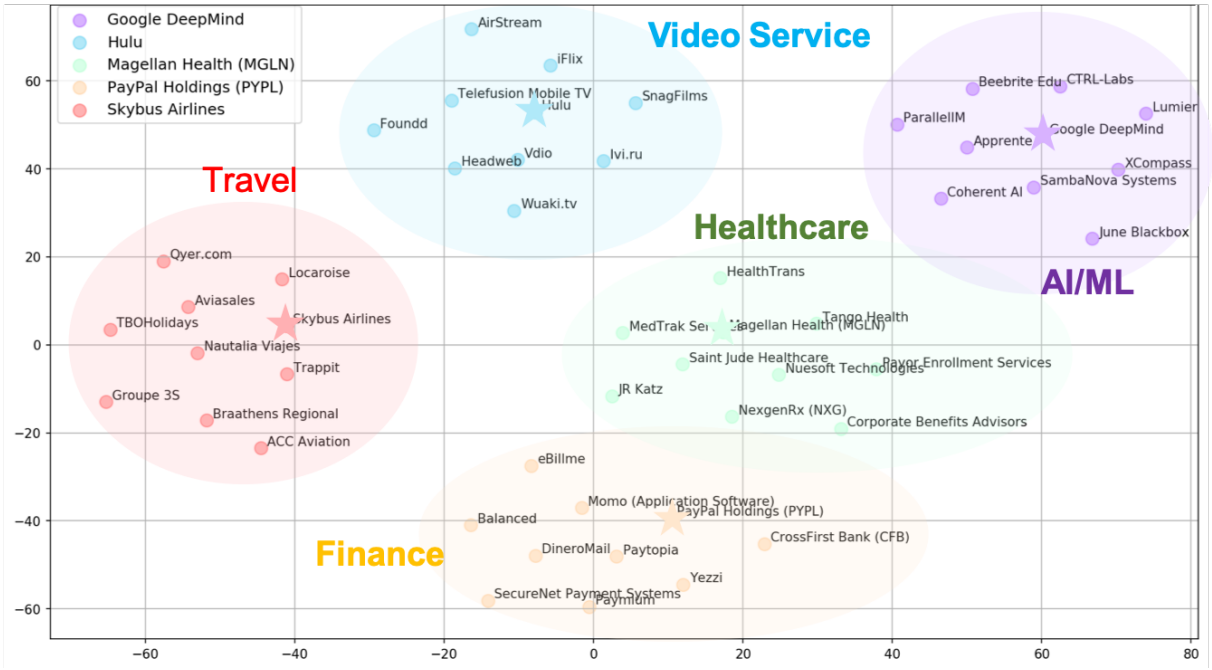

Figure 2: Visualization and Validation of Entity Embeddings with t-SNE

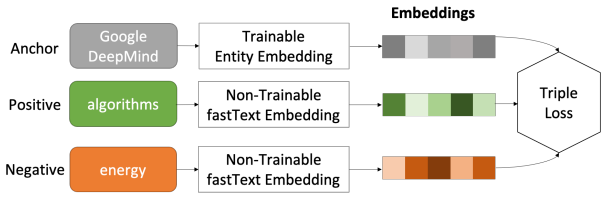

Figure 3: Model for Entity Embedding

between them and entity embeddings. We use LSTM to embed mentions from their context. Similar idea is adopted elsewhere (Kolitsas, Ganea, and Hofmann 2018), but instead of using a Bi-LSTM over the whole context, we apply two shorter LSTMs to embed mention from two directions (as shown in Figure 4), making the embedding more targeted with less parameters involved.

Given a mention m_t, we treat its left n words $\{w_{t-n}, ..., w_{t-1}, w_t\}$ (mention words included) and its left context, and right n words$\{w_t, w_{t+1}, ..., w_{t+n}\}$ as its right context (mention words included)[6]

$$\begin{aligned} h_t^l &= \overrightarrow{LSTM}(w_{t-1}^l, w_t) \\ h_t^r &= \overleftarrow{LSTM}(w_{t+1}^r, w_t) \end{aligned} \tag{3}$$

In addition to LSTM, we apply an attention layer to distinguish the influence of words. We multiply last layer's output from LSTM $\{x_i, .., x_j\}$ with attention weights, and get a context representation v.

$$\begin{aligned} \alpha_k &= < \mathbf{w}_\alpha, x_k > \\ a_k &= \frac{exp(\alpha_k)}{\sum_{s=i}^{j} exp(\alpha_s)} \\ g &= \sum_{k=i}^{j} a_k x_k \end{aligned} \tag{4}$$

Thus, we form a mention's vector by concatenating its left and right context representations, g_l and g_r:

$$\begin{aligned} g_m &= [g_l; g_r] \\ V_m &= FC(g_m) \end{aligned} \tag{5}$$

where FC is a fully connected feed-forward neural network. When we get the mention embedding V_m, given a pretrained entity embedding vector V_e, we can calculate similarity between these two vectors based on Euclidean distance.

$$D_{smc} = d(\mathbf{V_m}, \mathbf{V_e}) = \sqrt{\sum_{i=1}^{n} (V_{m_i} - V_{e_i})^2} \tag{6}$$

Contrastive Loss Function We combine both D_{syx} and D_{smc} as our target to train the model. The final distance is defined as:

$$D_W = \lambda_{syx} D_{syx} + \lambda_{smc} D_{smc} \tag{7}$$

Then, we apply a contrastive loss function to formulate our object loss function.

$$L = (Y)\frac{1}{2}(D_W)^2 + (1 - Y)\frac{1}{2}\{max(0, m - D_W)\}^2 \tag{8}$$

where Y is the ground truth value, where a value of 1 indicates that mention m and entity e is matched, 0 otherwise.

[6]We adopt the same pre-trained word embedding as for entity embedding.

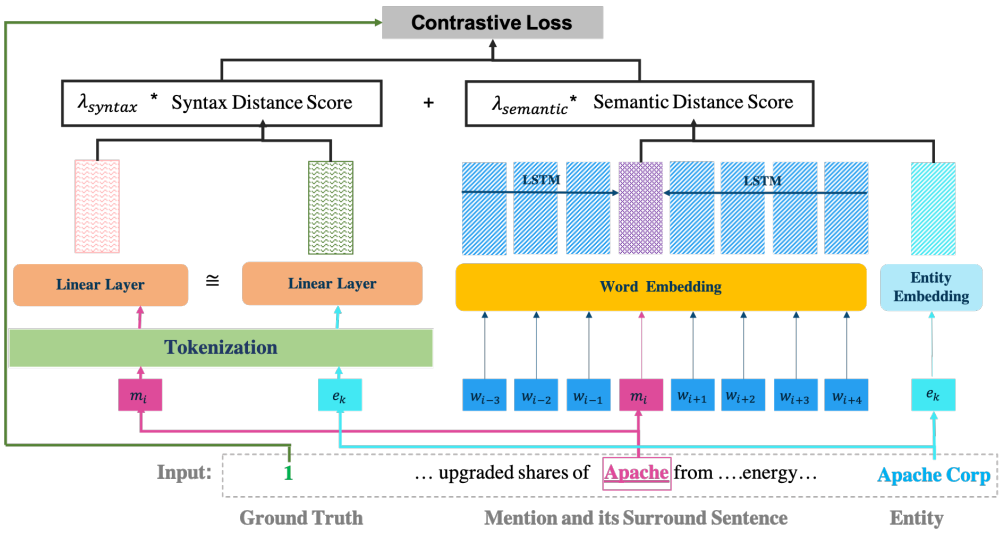

Figure 4: JEL Model Framework

Experiment and Analysis

Data Preparation

We first applied spaCy over financial news to detect all the named entity mentions. SpaCy features neural models for named entity recognition (NER). By considering text capitalization and context information, spaCy claims an accuracy above 85% for NER. Satisfied with spaCy's performance, we used it on financial news to recognize all the critical mentions that are tagged with "ORG". The data preparation process was as follows:

1) We extracted mentions from the financial news with spaCy.
2) We applied bi-gram cosine similarity between the extracted mentions and company names in our internal knowledge grpah.
3) If the similarity score between a mention string and an entity name is smaller than 0.5, we treated that as a strong signal that the two are not linked, and marked them as 0.
4) If the similarity score between a mention string and an entity name is equal to 1.0, we manually checked the list to avoid instances that two different entities share the same name (infrequent), and marked the pair as 1.
5) If a mention and an entity name have cosine similarity larger than 0.75, but smaller than 1.0, we manually labeled:
   (a) Some cases are easy to tell, such as: "Luminet" vs "Luminex", we labeled those instances as 0 directly.
   (b) Some cases can be decided according to their description/context. We printed mention's context information and entity's description respectively, and made the decision based on those texts, such as "Apple" vs "Apple Corps.".
   (c) Some other cases need help from publicly information found through internet search to decide, such as "Apollo Management" vs "Apollo Global Management".
6) If a mention and an entity name have cosine similarity between 0.5 and 0.75, we discarded it. These cases are too many for manual labeling, and too complicated for machine labeling.
7) Negative examples from step 3, make the dataset very imbalanced containing many more negative pairs. We counted the number of examples obtained in steps 4. and 5., and randomly sampled a comparable number from the examples gathered in step 3.

In total, we have labeled 586, 975 ground truth mention-entity pairs, with 293,949 positive mention-entity pairs, and 293, 026 negative pairs. We split 80% of the data as training data, 10% as validation data, and 10% as testing data.

Baselines

1) **String Matching** We chose Bi-Gram and Tri-Gram Cosine Similarity as two of baselines. Before similarity calculation, all tokens were weighted with tf-idf scores. We set 0.8 as the threshold.
2) **Context Similarity** We used Jaccard and Cosine similarity to measure the similarities between mention context and entity descriptions. A potential matched mention-entity pair should share at least one context word.
3) **Classification** We chose Logistic Regression (LR) and SVM for experiments. We adopted the feature engineering method defined in (Zheng et al. 2010), but only kept the following features that we can generate from our data:
   – StrSimSurface: edit-distance among mention strings and entity names.
   – ExactEqualSurface: number of overlapped lemmatized words in mention strings and entity names.
   – TFSimContext: TF-IDF similarity between mention's context and entity's description

	True Positivve	True Negative	False Positive	False Negative	Precision	Recall	F1-Sccore	Accuracy
JEL	0.5	0.4991	0.0009	0	0.9982	1	0.9991	0.9991
SVM-Rank	0.4826	0.4826	0.0174	0.0174	0.9652	0.9652	0.9652	0.9652
ENEL	0.3875	0.4922	0.0078	0.1125	0.9803	0.775	0.8656	0.8797
Tri-Gram Cosine	0.4423	0.4303	0.0666	0.0577	0.8691	0.8846	0.8768	0.8726
LR	0.4942	0.3712	0.1288	0.0058	0.7933	0.9884	0.8801	0.8654
SVM	0.4958	0.3589	0.1411	0.0042	0.7785	0.9916	0.8722	0.8547
Bi-Gram Cosine	0.4496	0.3794	0.1206	0.0504	0.7885	0.8992	0.8402	0.829
Jaccard Context	0.2444	0.3643	0.1357	0.2556	0.6430	0.4888	0.5554	0.6087
Cosine Context	0.4912	0.0116	0.4885	0.0088	0.5014	0.9824	0.6639	0.5028

Table 1: Performance Comparison via Precision and Recall

  - WordNumMatch: the number of overlapped lemmatized words between mention's context and entity's description.

4) **Learn to Rank** We used SVM-RANK as the representation of Learn to Rank. We adopted the same features as defined above.

5) **Deep Learning** We implemented state-of-the-art deep learning algorithm (Kolitsas, Ganea, and Hofmann 2018) (ENEL) as one of our baselines. In their original study, the authors utilized a Wikipedia derived entity embedding. But we utilized our own entity embedding for learning.

Comparison on Accuracy

We first compare the methods with *Precision* and *Recall*. For an easier comparison, we scaled each of *True Positive*, *True Negative*, *False Positive*, and *False Negative* into [0,0.5] showing as following.

- True Positive=$\frac{\text{Count(Predict=1 \& Truth=1)}}{2 \times \text{Count(Truth=1)}}$
- True Negative=$\frac{\text{Count(Predict=0 \& Truth=0)}}{2 \times \text{Count(Truth=0)}}$
- False Positive=$\frac{\text{Count(Predict=1 \& Truth=0)}}{2 \times \text{Count(Truth=0)}}$
- False Negative=$\frac{\text{Count(Predict=0 \& Truth=1)}}{2 \times \text{Count(Truth=1)}}$

The result is shown as Table 1, in which:

- Precision=$\frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$
- Recall=$\frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$
- F1-Score=$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
- Accuracy = True Positive+True Negative

From Table 1, we find context based methods perform poorly as expected. Descriptions in our knowledge graph have very different wording styles from financial news. Simply comparing context words will definitely result in low accuracy. SVM-Rank surprisingly outperforms ENEL. The reason here is that ENEL does not model character features properly. In SVM-Rank, we have carefully designed character features, (e.g., edit distance and tf-idf similarity), but ENEL just embeds 36 single character embeddings. This result also indicates that without good character learning, even deep learning could not solve the linking problem well.

	#parameters	# character features	processing time
JEL	10915800	151622	∼1min/batch
ENEL	11021400	36	∼20 min/batch

Table 2: Comparison among JEL and ENEL

JEL performs the best. We will mainly discuss the reason that JEL outperforms ENEL. First JEL involves more character features. ENEL just embeds 36 characters (26 letters + 10 digits), but JEL computes 151622 character features (as shown in Table 2). This configuration supports JEL with a better performance in capturing character patterns. For example, JEL could successfully link "Salarius Pharm LLC" to "Salarius Pharmaceuticals" but ENEL missed this link. Second, ENEL jointly embeds all characters and words from context and mention itself into a mention's vector, and minimizes the distance between this mention vector and a pre-trained entity embedding vector. However, the entity embeddings themselves are generated without character information (Ganea and Hofmann 2017). Character embeddings in ENEL, especially character embeddings from context words, somehow add noise to semantic embeddings, and impact final performance. In addition, Table 2 gives a brief overview of efficiency comparison between JEL and ENEL. Although JEL and ENEL share similar number of parameters, JEL trains faster than ENEL. JEL utilizes linear layers to learn character patterns, which is easier to learn than an embedding layer in ENEL.

Comparison on Precision

Accuracy can only check a method's ability in distinguishing positive samples from negative samples. In a real EL task, we care more about a method's ability in finding the correct entity for a given mention. In this section, we utilize the "Precision at top K (P@K)" to compare the methods (shown in Figure 5 with $K \in \{1, 5, 10\}$). LR, SVM, and SVM-Rank get 0s at all P@Ks. Both LR and SVM are classifiers, which are good at distinguishing positive pairs from negative pairs. However, they may label multiple mention-

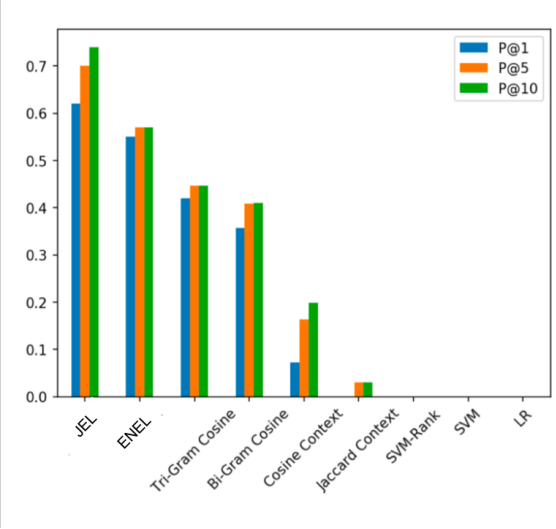

Figure 5: Performance Comparison via P@K(Precision at top K)

entity pairs as positive, but rank false positive pairs ahead. The SVM-Rank's failure is surprising. We believe the reason lies in the training data. We only have 1 and 0 as labels, but no specific ranking order, which is hard to obtain for a $0-1$ problem, to train the model solidly. Without a ranking ground truth, SVM-Rank fails in our task. For the other four methods, we rank entities based on distance or similarity. JEL still achieves the best performance.

Deployment and Business Impact

JPMC has built a large-scale knowledge graph for internal use, that integrates data from third party providers with its internal data. The system contains several million entities (e.g. suppliers, investors, etc.) and several million links (e.g. supply chain, investment, etc.) among those entities [7]. Entity linking is one of the core problems that needs to be solved when ingesting unstructured data. Currently, we are working closely with business units to process incoming news articles to extract news about companies, and connect them to the corresponding entities in the knowledge graph. Such news analytics will support users across JPMC in discovering relevant and curated news that matter to their business.

Here we present a concrete example of the use of such news analytics. To protect the customer cofidentiality, the actual name of the company has been changed, but the illustrated computation is the same. "Acma Retail Inc" filed for bankruptcy due to the pandemic, and a lot of JPMC clients could feel stress as they are suppliers to Acma. Such stress can pass deep down into its supply chain and trigger financial difficulties for other clients. JPMC may face different levels of risks from suppliers with different orders in Acma's supply chain. With "Acma" mentioned in financial news linked with "Acma Global Retail Inc" in our knowledge graph (distinguished from "Acma Furniture, LLC", "Acma Enterprise System", etc.), we can accurately track down Acma supply chain, identify stressed suppliers with different revenue exposure, and measure our primary risk due to Acma's bankruptcy. Once stressed clients with significant exposure are detected, alerts will be sent out to corresponding credit officers. If "Acma" was linked with incorrect entities, it will result in too many false signals, resulting in wasted effort.

A similar news analytics system, SNOAR, has been previously reported (Goldberg et al. 2003). SNOAR used a rule based and fuzzy match techniques for entity linking, which are not able to handle the example illustrated in Figure 1. In fact, before JEL was developed, *Tri-Gram Cosine Similarity* was tested for name string matching. As shown in Table 1 and Figure 5, tri-gram cosine similarity has practical limitations and could not distinguish entities sharing similar names (as shown in Figure 1). Compared to fuzzy match or tri-gram cosine similarity, JEL provides a better and more controllable solution.

In the deployed version of JEL, we will apply another blocking layer (overlap blocker for current configuration) ahead of JEL to reduce the candidate volume for each mention in an article. Entities sharing less than 2 bi-gram tokens will be filtered out in the blocking stage, and the rest of candidate entities will be sent to JEL for a more sophisticated linking proccess. This blocking process will dramtically reduce computational cost.

During the first-round deployment, we are incorporating JEL into a streaming news platform, and its online performance will be tested on indicators including:

- *Scope*: Ability in linking all available JPMC clients
- *Timeliness*: Ability in providing near real-time alerts
- *Accuracy*: Ability in sending accurate alerts
- *Flexibility*: Ability in integration of various components for end-to-end delivery of solutions.

Our first step is to collect user feedbacks for parameter fine-tuning and algorithm re-configuration. Once the initial deployment is successful, we will release JEL as a standalone and reusable component with an API to provide a firm-wide service that can be used in various applications.

Conclusion

Knowledge Graphs are becoming a mission critical technology across many industries. Within JPMorgan Chase, we are using a knowledge graph as a company-wide resource for tasks such as risk analysis, supply chain analysis, etc. A core problem in utilizing this knowledge is EL. Existing EL models did not generalize to our internal company data, and therefore, we developed a novel model, JEL, that leverages margin loss function and deep and wide learning. Through an extensive experimentation, we have shown the superiority of this method. We are currently in the process of deploying this model on a financial news platform within our company. Even though our testing was done on our internal data, we believe, that our approach can be adapted by other companies for entity linking tasks on their internal data.

[7] For proprietary reasons, we cannnot reveal the exact number of nodes and links

References

Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; and Shah, R. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, 737–744.

Ceccarelli, D.; Lucchese, C.; Orlando, S.; Perego, R.; and Trani, S. 2013. Learning relatedness measures for entity linking. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 139–148.

Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, 7–10.

Cucerzan, S. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 708–716.

Eshel, Y.; Cohen, N.; Radinsky, K.; Markovitch, S.; Yamada, I.; and Levy, O. 2017. Named entity disambiguation for noisy text. *arXiv preprint arXiv:1706.09147* .

Francis-Landau, M.; Durrett, G.; and Klein, D. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. *arXiv preprint arXiv:1604.00734* .

Ganea, O.-E.; and Hofmann, T. 2017. Deep joint entity disambiguation with local neural attention. *arXiv preprint arXiv:1704.04920* .

Goldberg, H. G.; Kirkland, J. D.; Lee, D.; Shyr, P.; and Thakker, D. 2003. The NASD Securities Observation, New Analysis and Regulation System (SONAR). In *IAAI*, 11–18.

Han, X.; Sun, L.; and Zhao, J. 2011. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 765–774.

He, Z.; Liu, S.; Li, M.; Zhou, M.; Zhang, L.; and Wang, H. 2013. Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 30–34.

Huang, H.; Heck, L.; and Ji, H. 2015. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *arXiv preprint arXiv:1504.07678* .

Kolitsas, N.; Ganea, O.-E.; and Hofmann, T. 2018. End-to-end neural entity linking. *arXiv preprint arXiv:1808.07699* .

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* .

Martins, P. H.; Marinho, Z.; and Martins, A. F. 2019. Joint learning of named entity recognition and entity linking. *arXiv preprint arXiv:1907.08243* .

Mihalcea, R.; and Csomai, A. 2007. Wikify! Linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 233–242.

Mikolov, T.; Yih, W.-t.; and Zweig, G. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, 746–751.

Milne, D.; and Witten, I. H. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, 509–518.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: A Core of Semantic Knowledge. In *16th International Conference on the World Wide Web*, 697–706.

Sun, Y.; Lin, L.; Tang, D.; Yang, N.; Ji, Z.; and Wang, X. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Twenty-fourth international joint conference on artificial intelligence*.

Varma, V.; Pingali, P.; Katragadda, R.; Krishna, S.; Ganesh, S.; Sarvabhotla, K.; Garapati, H.; Gopisetty, H.; Reddy, V. B.; Reddy, K.; et al. 2009. IIIT Hyderabad at TAC 2009. In *TAC*.

Yamada, I.; Shindo, H.; Takeda, H.; and Takefuji, Y. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343* .

Zhang, W.; Su, J.; Tan, C. L.; and Wang, W. T. 2010. Entity linking leveraging: automatically generated annotation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, 1290–1298. Association for Computational Linguistics.

Zheng, Z.; Li, F.; Huang, M.; and Zhu, X. 2010. Learning to link entities with knowledge base. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 483–491. Association for Computational Linguistics.