Convolutional Neural Network Based Semantic Tagging with Entity Embeddings

Asli Celikyilmaz
Microsoft
Silicon Valley
Mountain View, CA
asli@ieee.org

Dilek Hakkani-Tur
Microsoft Research
Silicon Valley
Mountain View, CA
dilek@ieee.org

Abstract

Unsupervised word embeddings provide rich linguistic and conceptual information about words. However, they may provide weak information about domain specific semantic relations for certain tasks such as semantic parsing of natural language queries, where such information about words or phrases can be valuable. To encode the prior knowledge about the semantic word relations, we extended the neural network based lexical word embedding objective function by incorporating the information about relationship between entities that we extract from knowledge bases [1]. In this paper, we focus on the semantic tagging of conversational utterances as our end task and we investigate two different ways of using these embeddings: as additional features to a linear sequence learning method, Conditional Random Fields (CRF), and as initial embeddings to a convolutional neural networks based CRF model (CNN−CRF) with shared feature layers and globally normalized sequence modeling components. While we obtain an average of 2% improvement in F-score compared to the previous baselines when the enriched embeddings are used as additional features for CRF models, we obtain slightly more gains - when the embeddings are used as initial word representations for the CNN-based CRF models.

1 Introduction

One of the strongest trends in speech and language processing at the moment is the use of word embeddings, which are vectors whose relative similarities correlate with semantic similarity. Such vectors are used both as an end in itself (for computing similarities between terms), and as a representational basis for downstream tasks like text/utterance classification, document clustering, part of speech tagging, named entity recognition, sentiment analysis, slot tagging, and so on. In this paper, we focus on the sequence learning problem, specifically semantic slot tagging in conversational understanding tasks. We show different ways of using knowledge graphs to enrich the entity embeddings and provide these embeddings to two sequence learning methods in different ways to improve semantic tagging performance.

Semantic tagging is crucial in recognizing words of semantic importance in a given natural language query such as:

	character		year	genre	type	name
who played	*zeus*	in the	*2010*	*action*	*movie*	*Titans*

The recognized semantic tags are used to send queries to the database to fetch relevant data so as to generate appropriate system response. Common approaches to building semantic taggers for labeled data are sequence learning algorithms such as conditional random fields (CRF) [2], recurrent

neural networks, that depend on large amounts of manually annotated data to achieve good performance. These models mainly focus on short-range syntactic dependencies, considering words in a user defined window. To better capture the long range dependencies previous work introduced sampling based inference methods [3], local classification models [4], additional loss functions in the objective function in Augmented Loss Framework [5], long short term memory approach incorporated into recurrent neural networks [6], etc. Unsupervised word embeddings can provide sequence models with rich linguistic and conceptual information about words [7, 8, 9], but, they may provide weak or no information about the domain specific semantic relations. Strengthening the long-range dependencies between words (e.g., *zeus* and *Titans*) is important for disambiguation of entities in queries. This, if true, is a useful property as it can help to improve other tasks such as classification, clustering, ranking, to name a few.

Our end goal is to use the enriched embeddings on a semantic tagging task, namely slot tagging in natural language understanding. In this paper, we investigate two different approaches: (1) using enriched embeddings directly as features to a `CRF` model (similar to [9]) and (2) injecting them as initial embeddings to a Convolutional Neural Network (`CNN`) based `CRF` model (`CNN-CRF`)[15]. In our earlier work [1], we have shown that when we use the enriched word embeddings as features for `CRF` models, we achieve up to 2% improvement in F-score in comparison to the baselines. In this paper, using them as prior embeddings to convolutional neural network models, we show further improvements in the performance (slightly more than the best performing `CRF` models). Empirical analysis of sequence learning methods (linear or non-linear) as a downstream method demonstrates the effect of enriched relational embeddings in capturing long term dependencies between words.

2 Learning Word Embeddings with Priors

We begin with first reviewing `word2vec` and some of the earlier work that extends its objective function to inject prior knowledge.

`Word2Vec` [14]. It is an efficient neural network language model. The algorithm takes unstructured text and learns embeddings ("features") about each word represented as a set of latent variables. These embeddings capture different relationships - both semantic and syntactic, allowing for some (very basic) algebraic operations, like "king-man+woman≅queen". The objective function, learns representations of each word w_t to maximize the log likelihood of each token given its context, namely, neighboring words within window size c:

$$\max \frac{1}{T} \sum_{t=1}^{T} \log p(w_t | w_{t-c}^{t+c}) \tag{1}$$

where w_{t-c}^{t+c} is the set of words in the window of size c centered at w_t (w_t included). Using the continuous bag of words model (`CBOW`) of word2vec, $p(w_t|w_{t-c}^{t+c})$ predicts the current word based on the context as follows:

$$p(w_t | w_{t-c}^{t+c}) := \frac{\exp\left({e_{w_t}^y}^T \cdot \sum_{-c \le j \le c, j \neq 0} e_{w_{t+j}}\right)}{\sum_w \exp\left({e_w^y}^T \cdot \sum_{-c \le j \le c, j \neq 0} e_{w_{t+j}}\right)} \tag{2}$$

In Eq. (2) e_w and e_w^y represent input and output embeddings, the scalar vector representations of each word w.

Relational Constrained Model (`RTM`) [13]. Learns embeddings to predict one word from another related word. Suppose we are given a list of synonymous or paraphrases of *N* words based on a knowledge source (e.g., Wordnet). `RTM` learns the word embeddings based on the paraphrase relation between the words. Thus, they introduce priors as paraphrases encoding synonymy relations such as "*analog*"∼"*analogue*" or "*bolt*"∼"*screw*". They change the objective function of the `word2vec` by dropping the context and learn the embeddings on the paraphrase data. The objective to maximize is the sum over all the words in the vocabulary, which is similar to Eq.(1) without the context:

$$\max \frac{1}{N} \sum_{i=1}^{N} \sum_{w \in R_{w_i}} \log p(w | w_i) \tag{3}$$

where $p(w|w_t)$=$\exp({e_w^y}^T e_{w_i}) / \sum_w \exp({e_w^y}^T e_{w_i})$. This model enables learning of embeddings such that they are predictive of related words in the resource. e_w and e_w^y are again input and output embeddings.

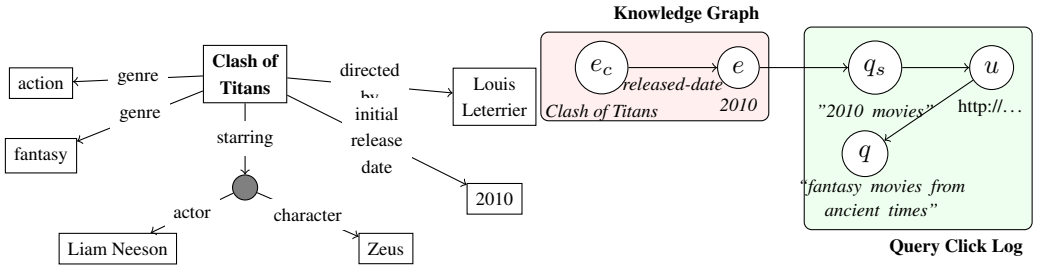


Table 1: (Left) A sub-graph from freebase with central entity *Clash of Titans* the movie. (Right) Query mining starting from seed knowledge base entities e_c, then searching the query click logs and later collecting related queries that share the same clicked URL.

`Joint` Model [13]. While `CBOW` learns lexical word embeddings from provided text, the `RTM` learns embeddings of words based on their similarity to other words provided by a knowledge resource. The `Joint` model combines the two through linear combination:

$$\max \frac{1}{T}\sum_{t=1}^{T}\log p(w_t|w_{t-c}^{t+c}) + \frac{C}{N}\sum_{i=1}^{N}\sum_{w\in R_{w_i}}\log p(w|w_i) \tag{4}$$

The left hand-side of the objective function (`CBOW`) learns the embeddings on the unstructured text, while the right hand-side (`RTM`) uses the paraphrase data. The `Joint` balances the two by a regularization parameter C (e.g., C=$\frac{1}{12}$ helps to allow `CBOW` to benefit from unlabeled data, but refine embeddings constraint by the paraphrase information.)

3 Mining Entities, Relations and Queries

We focus on the entity relations that are present in knowledge graphs, such as Freebase, which encode factual world knowledge in triples <*s*,*r*,*o*> of a pair of entities and their relations. Since our ultimate goal is semantic tagging of natural language queries in movies domain, we start by mining knowledge graph entities related to the movies domain.

• **Mining entities and relations related to domain (Triplets Data).** We start by listing all entities of *central type*, e_c (e.g. Freebase `film.film` type for the movie domain). Then, for each entity, we collect other entities e that has incoming relation from e_c (e.g., e=*2010* via the relation *release-date*). We automatically formulated realizations of triplets from each e_c and its related e's and their relations, r (e.g., <s_i,r_i,o_i>=<*Clash of Titans*, *released*, *2010*>) using the methods as in [16].

• **Mining Query-Entity-Type (QET) relational data for training embeddings.** We mine queries related to each entity e from the query click logs as sketched in Figure **??**. We automatically generate seed queries q_s for each e (e.g., *2010* becomes "*2010 movies*" or "*fantasy films recent releases*") and search q_s in click logs and their clicked URL u. We collect other queries q that also link to the URLs u to effectively perform two-step random walk on the click graph [17]. We obtain list of query q, its related entity e and the entity id triplets (e.g., <*q,e,id*>), which we call QET relational data (e.g., <*"show me 2010 movies","2010"*, freebase.com/mt/09fu66>). (see Table 3 Right). For detailed information about our data collection, please refer to [16].

4 Enriching Word Embeddings

4.1 Context and Entity Constrained Model - `CECM`

The first approach, the context constrained model (`CECM`), uses query-entity pairs as training data and learns representation for each word w_t by *implicitly* constraining the context of the word with the corresponding entity. The `CECM` maximizes the log likelihood of each query token t given its context words within a window size of c along with the entity e related to the query:

$$\max \frac{1}{T}\sum_{t=1}^{T}\log p(w_t|w_{t-c}^{t+c}, e) \tag{5}$$

4.2 Relation Encoding Model - REM

Given triplets S=$\{<s_i,r_i,o_i>\}$, i=1,...$|S|$ of relations extracted from knowledge graph, we can assume that the functional relation induced by the r-labeled arcs should correspond to a translation of the embeddings. Thus, we can enforce that **s+r**≈**o** when $<s,r,o>$ holds. This is similar to saying that given the relational information, the '*Clash of Titans*'+'*released*'≈'*2010*'. We encode the relations between s+r and o similar to the Joint model of [13]. We define **R** as a set of relations R' between s+r, and object o extracted from knowledge graph (e.g., R'=*released-date* where the realizations are *r*={*released*, *debuted*, *launched*,...}). Our new relation encoding model (REM) maximizes the log probability as:

$$\frac{1}{T}\sum_{t=1}^{T}\log p(w_t|w_{t-c}^{t+c},e)+\sum_{R'\in\mathbf{R}}\frac{1}{N}\sum_{i=1}^{N}\sum_{s+r\in R'_{o_i}}\log p(s+r|o_i) \tag{6}$$

The left-handside of Eq. (6) is our CECM model, which trains on the QET dataset to learn the lexical word embedding with the entity types of the query in its context. The right hand-side (the relational part) optimizes the relational word embeddings for each relation $R'\in\mathbf{R}$ separately and trains on the list of triplets. The log probability is computed as:

$$p(s+r|o_i):=\exp({e^y_{s+r}}^T e_{o_i})/\sum_{s+r}\exp({e^y_{s+r}}^T e_{o_i}). \tag{7}$$

As an intermediate step, at each epoch, we calculate the output embeddings e^y_{s+r} by algebraically adding the output embeddings of the subject s and relation r predicted by the the CECM model as follows: e^y_{s+r}=e^y_s+e^y_r. Note that, the entities and relations in the triplets data may comprise of the unigrams as well as *n*-gram words.

5 Semantic Tagging with Enriched Entity Embeddings

Once we learn the enriched embeddings of the words and entities, we use them to improve the slot tagging performance of conversational understanding task. We summarize the two approaches related to our work in this paper:

5.1 Enriched Embeddings used as Features

We convert the scalar valued vector representations of words into class-based representation similar to [7, 9]. Such word classes are used as additional features [9] for the CRF models .

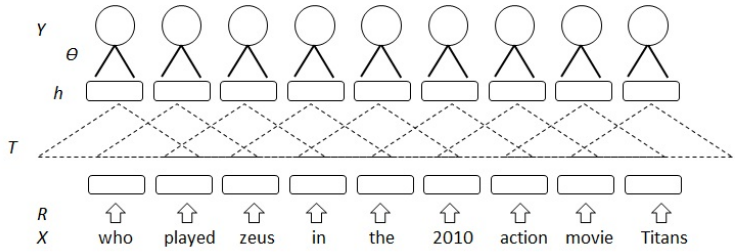
5.2 Enriched Embeddings used as Prior Representations

NN based CRF has been investigated in the literature. In [19, 20], conditional neural field and neural CRF were introduced which combined the NN based feature learning and the CRF models. Similar approaches have also been used for phone recognition with success [21, 22].

In contrast to the above approaches, in [15] the sequence labeling is handled as a full blown CRF model, introducing another CNN-based CRF model (CNN−CRF), that stacks the convolutional and sequence tagging layers. Simply put, the first layer is a convolutional neural network, where the sentence representations are learnt and the top layer is nothing but a CRF model, as shown in Figure 1. Due to the fact that this model takes a globally normalized conditional approach (similar to CRF models), in this paper, we use this method to benchmark our results against CRF models.

In CNN−CRF no normalization is performed at each individual position. In fact, the top layer is essentially the same as a first order CRF model. The only difference is that the features from the word sequence are automatically extracted continuous-valued features, instead of predefined indicator functions. As shown in Figure 1, the word sequence is represented as the concatenated word vectors (*R*). Such word vectors may also encode positional information indicating which word in the sequence the tag will be predicted for. The feature transform *T* spans over a fixed-sized n-gram window and slides over the input sequence. The total score of the sequence is factorized into the sum of scores at each word position i. $t(Y_{i-1},Y_i)$ is the tag transition score from Y_{i-1} to Y_i. The

Figure 1: CNN−CRF: Globally normalized conditional model based on CNN. The top layer is essentially the same as CRF



$h_{ij}(X_i, R, T)$ denotes the *j*th element in the feature vector extracted out of the n-gram window centered at X_i, and $\theta_j(Y_i)$ is the corresponding feature weight associated with the tag Y_i. The bottom layers can be trained using the well-known back propagation algorithm. The derivative of the loss function is taken with respect to each $h_{ij}(X_i, R, T)$ at the top layer, and obtain the derivative with respect to *T* and *R* by applying the chain rule.

In this paper, we use the enriched embeddings to represent the word vectors (*R*). Speifically, instead of using random initialization (e.g., samples from Gaussian distribution), we deterministically set the initial embedding to be the enriched embeddings learnt in section 4.

6 Experiments

Datasets. To evaluate our approach we use the movie domain dataset from [25], available for download. The focus here is on audiovisual media in the movie domain. The user interacts via voice with a system that can perform a variety of tasks such as browsing, searching, etc. We used crowd-sourcing to collect and annotate queries with semantic entity. The data contains 3,338 training and 1,084 test queries. Each query is tagged with 25 different semantic entities, e.g., `movie-director` ('*James Cameron*'), `movie-title` ('*Die Hard*'), `genre` ('*feel good*'), `description` ('*black and white*', '*pg 13*'), `review-rate` ('*epic*', '*not for me*'), `theater-location` ('*near me*','*city center*'), etc.

We have collected around 250K different movie names and 10 million related queries. To train the word embeddings, after cleanup and filtering the mined data, we compiled around 1M movies related query-entity pairs to construct the QET dataset, using the intuition presented in the previous section. We extracted ∼100K entity-relation triplets to construct the triplets datasets for each relation.

Baseline Embedding Models: We train the baseline embeddings, i.e., `CBOW` on just the 1M movie queries from the QET dataset (excluding the entity and type information). For training the baseline `RTM` and `Joint` models [13] we use the entity lists as paraphrase data. Specifically, using the entity lists from QEC dataset, we randomly selected entities from each entity type to construct paraphrase pairs (e.g., using `release-date` entity type, we add an entry to the paraphrase file as "1960" ≈"2003").

Proposed Embedding Models: For training the first of our new embeddings models, `CECM`, we use the QET dataset, encoding the entity types *implicitly* at training time. For the `REM` models, we use the entity relation triplet pairs representing the relational data as well as the QET dataset.

Features for `CRF` models: We convert the scalar valued vector representations of words into class-based representation similar to [7, 9]. Following [9], we use the classes to generate class-based features for the `CRF` models . If a query contains a compound entity (with '_'), we assign each word of the compound the same class as the compound (e.g., if we have class("*Clash_of_Titans*")=c876, then we assign class("*Clash*")=c876, class("*of*")=c876, class("*Titans*")=c876).

Features for `CNN−CRF` models: We used the learnt features as is and injected into the convolutional layer of the `CNN−CRF`.

Evaluations and Results on Semantic Tagging. We use `CRF` to build semantic tagging models and add several embedding features from baseline and the models that enrich the embeddings as follows:

	Model	F-Score on Test
Baselines	CRF-1	86.15
	CRF-2	86.09
	CNN-CRF	86.94
Baselines with Embeddings	CRF-CBOW	86.61
	CRF-RTM	87.58
	CRF-Joint	87.65
	CNN-CRF-CBOW	86.80
	CNN-CRF-RTM	87.97
	CNN-CRF-Joint	88.41
New with Enriched Embeddings	**CRF-CECM**	**88.58**
	CRF-REM	88.12
	CRF-(CECM->REM)	**88.70**
	CNN-CRF-CECM	87.46
	CNN-CRF-REM	88.17
	CNN-CRF-(CECM->REM)	**88.79**

Table 2: Results for semantic tagging on the test dataset with embedding features from different models.

• CRF: Baseline models consider models with 1-gram (CRF-1) and 2-gram (CRF-2) word features with a window of 5 words, without the embedding features.
• {CRF-CBOW, CRF-RTM, CRF-Joint}: Baseline models using embedding features from CBOW, RTM, and Joint models respectively.
• {CRF-CECM, CRF-REM, CRF-(CECM->REM)}: These models use embedding features from CECM, REM. The last one uses embedding features obtained from the REM models using the CECM output as initial embeddings.

Similarly, we use CNN-CRF to build semantic tagging models and use several embedding features as prior embedding values to build the following models:
• CNN-CRF: Baseline models consider models with a window of 5 words, where the word vectors are initialized based on random samples from Gaussian distribution.
• {CNN-CRF-CBOW, CNN-CRF-RTM, CNN-CRF-Joint}: Baseline models using word embedding features from CBOW, RTM, and Joint models respectively.
• {CNN-CRF-CECM, CNN-CRF-REM, CNN-CRF-(CECM->REM)}: These models use embedding features from CECM, REM. The last one uses embedding features obtained from the REM models using the CECM output as initial embeddings.

We present the empirical results in Table 2. For both learning methods - regardless of the way the embeddings are used in sequence tagging, we obtain gains over the baseline embeddings (non-enriched). Our analysis has let us conclude that using the explicit relational embeddings from Knowledge Graphs (e.g., Freebase in our case) can boost the implicit embeddings and together they are better suited for semantic tagging tasks. We observe that CNN-based CRF models are performing slightly better than the regular CRF models. In particular, the CNN-CRF-(CECM->REM) yields the best performance among all the other models. As a future work, we would like to obtain the enriched embedding for other domains, such as restaurants and transportation, in which precision in resolution of entities are also important for a better user experience in human to machine conversational dialogs.

In addition, we would like to investigate on improvements for the CNN models by implementing recent techniques such as drop-out [26], word-hashing [27], to name a few.

7 Conclusion

We empirically investigated difference usages of domain specific word embeddings on sequence learning task. We learn that the embeddings of a word can be enriched based other words that together have functional relations with. The enriched embeddings provide rich semantic information to the semantic tagging tasks, proving information about the long term relations. Although the results indicate that the type of the sequence learning algorithm has little effect on the performance of the sequence learning models, we observed a slight improvement when the convolutional neu-

ral network based sequence learning models are used. As a future work, we will conduct similar experiments on different domains to validate that our findings are broadly applicable.

Acknowledgments

The authors would like to thank Puyang Xu (Microsoft) and Ruhi Sarikaya (Microsoft) for useful dicussions and help on running the convolutional neural network software tool.

References

[1] A. Celikyilmaz, D. Hakkani-Tur, P. Pasupat, and R. Sarikaya. Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems. 2015.

[2] J.D. Lafferty, A. McCallum, and F.C.N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2011.

[3] J. Finkel, C.D. Manning, and A.Y. Ng. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. 2006.

[4] P. Liang, H. Daume-III, and D. Klein. Structure compilation: trading structure for features. 2008.

[5] H. Daume-III, J. Langford, and D. Marcu. Search-based structured prediction. 2009.

[6] J. Martens and I. Sutskever. Solving the problem of cascading errors: Approximate bayesian inference for lingusitic annotation pipelines. 2011.

[7] M. Bansal and and K. Livescu K. Gimpel. Tailoring continuous word representations for dependency parsing. In *Proc. of ACL*, 2014.

[8] J. Guo, W. Che, H. Wang, and T. Liu. Revisiting embedding features for simple semi-supervised learning. In *Proc. of EMNLP*, 2014.

[9] R. Sarikaya, A. Celikyilmaz, A. Deores, and M. Jeong. Shrinkage based features for slot tagging with conditional random fields. In *Proc. of Interspeech*, 2014.

[10] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. Learning sentiment-specific word embedding for twitter sentiment classification. 2014.

[11] R. Lebret, J. Legrand, and R. Collobert. Is deep learning really necessary for word embeddings ? 2013.

[12] J. Weston, A. Bordes, O. Yakhnenko, and N. Usunier. Connecting language and knowledge bases with embedding models for relation extraction. 2013.

[13] M. Yu and M. Dredze. Improving lexical embeddings with semantic knowledge. 2014.

[14] T. Mikolov, K. Chen I. Sutskever, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. 2013.

[15] P. Xu and R. Sarikaya. Convolutional neural network based triangular crf for joint intent detection and slot filling. 2014.

[16] P. Pasupat and D. Hakkani-Tur. Unsupervised relation detection using automatic alignment and query patterns extracted from knowledge graphs and query click logs. 2015.

[17] N. Craswell and M. Szummer. Random walks on the click graph. 2007.

[18] Antoine Bordes, Nicolas Usunier, Alberto Garca-Durn, Jason Weston, and Oksana Yakhnenko. Irreflexive and hierarchical relations as translations. *CoRR*, pages –1–1, 2013.

[19] L. Bo J. Peng and J. Xu. Conditional neural fields. 2009.

[20] T.M.T. Do and T. Artieres. Neural conditional random fields. 2010.

[21] D. Yu A. Mohamed and L. Deng. Investigation of full sequence training of deep belief networks for speech recognition. 2010.

[22] G. Andrew and J. Bilmes. Sequential deep belief networks. 2012.

[23] S. Wang D. Yu and L. Deng. Sequential labeling using deep-structured conditional random fields. 2010.

[24] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. 2008.

[25] D. Hakkani-Tur, A. Celikyilmaz, L. Heck, and G. Tur. Probabilistic enrichment of knowledge graph entities for relation detection in conversational understanding. 2014.

[26] P. Xu and R. Sarikaya. Targeted feature dropout for robust slot filling in natural language understanding. 2014.

[27] Y. Shen, X. He, J. Gao, and G. Mesnil L. Deng. A latent semantic model with convolutional-pooling structure for information retrieval. 2014.