Enriching Word Embeddings Using Knowledge Graph for Semantic Tagging in Conversational Dialog Systems

Asli Celikyilmaz
Microsoft
asli@ieee.org

Dilek Hakkani-Tur
Microsoft Research
dilek@ieee.org

Panupong Pasupat
Stanford University
ppasupat@cs.stanford.edu

Ruhi Sarikaya
Microsoft
ruhi.sarikaya@microsoft.com

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 can be valuable. To encode the prior knowledge about the semantic word relations, we present new method as follows: we extend the neural network based lexical word embedding objective function (Mikolov et al. 2013) by incorporating the information about relationship between entities that we extract from knowledge bases. Our model can jointly learn lexical word representations from free text enriched by the relational word embeddings from relational data (e.g., Freebase) for each type of entity relations. We empirically show on the task of semantic tagging of natural language queries that our enriched embeddings can provide information about not only short-range syntactic dependencies but also long-range semantic dependencies between words. Using the enriched embeddings, we obtain an average of 2% improvement in F-score compared to the previous baselines.

Introduction

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

$$\textit{who played the } \overbrace{\textit{zeus}}^{\texttt{character}} \textit{ in the } \overbrace{\textit{2010}}^{\texttt{year}}\ \overbrace{\textit{action}}^{\texttt{genre}}\ \overbrace{\textit{movie}}^{\texttt{type}}\ \overbrace{\textit{Titans}}^{\texttt{name}} ?$$

The recognized semantic tags are used to form queries to the database to fetch relevant data and to generate appropriate system response. Common approaches to building semantic taggers are sequence learning algorithms such as conditional random fields (CRF) (Lafferty, McCallum, and Pereira 2011), 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 (Finkel, Manning, and Ng 2006), local classification models (Liang, Daume-III, and Klein 2008), additional loss functions in the objective function in Augmented Loss Framework (Daume-III, Langford, and Marcu 2009), long short term memory approach incorporated into recurrent neural networks (Martens and Sutskever 2011), etc. Unsupervised word embeddings can provide sequence models with rich linguistic and conceptual information about words (Bansal and K. Gimpel 2014; Sarikaya et al. 2014), but, they may provide weak or no information about the domain specific semantic relations. Strengthening the long-range dependencies (e.g., *zeus* and *Titans*) is important for disambiguation of entities in queries.

Adapting word embeddings, such as jointly capturing syntactic and semantic information, can further enrich semantic word representations for several tasks, e.g., sentiment analysis (Tang et al. 2014), named entity recognition (Lebret, Legrand, and Collobert 2013), entity-relation extraction (Weston et al. 2013), etc. (Yu and Dredze 2014) has introduced a lightly supervised word embedding learning extending `word2vec`-a neural network language model for learning continuous word representations (Mikolov et al. 2013). They incorporate prior information to the objective function as a regularization term considering synonymy relations between words from Wordnet (Fellbaum 1999).

In this work, we go one step further and investigate if enriching the `word2vec` word embeddings trained on unstructured/unlabeled text with domain specific semantic relations obtained from knowledge sources (e.g., knowledge graphs, search query logs, etc.) can help to discover relation aware word embeddings. Unlike earlier work, we encode the information about the *relations* between phrases, thereby, entities and relation mentions are all embedded into a low-dimensional vector space. We introduce two new methods for injecting entity types and their relations to jointly learn the enriched word embeddings on millions of search query logs. When we use the enriched word embeddings as features for semantic tagging models, we achieve up to 2% improvement in F-score in comparison to the baselines.

Learning Word Embeddings with Priors

We begin by reviewing the `word2vec` and earlier work that extends its objective function to inject prior knowledge.

`Word2Vec` (Mikolov et al. 2013). 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

Copyright © 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

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:

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

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:

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

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) (Yu and Dredze 2014). 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:

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

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.

Joint Model (Yu and Dredze 2014). 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:

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

The left hand-side of the objective function (CBOW) learns the embeddings on the unstructured text, while the right hand-side (RCM) 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.)

Mining Entities, Relations and Queries

Before we explain our enriched word embedding learning models that incorporate entities and their relations, we briefly describe our approach for collecting unsupervised queries, entities and relations for training the embeddings.

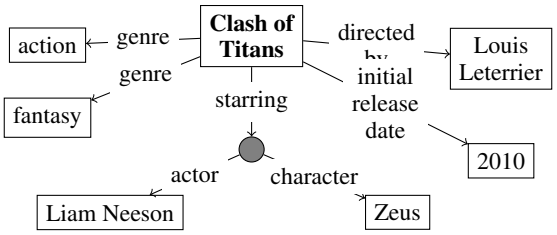

Figure 1: A sub-graph from freebase with central entity *Clash of Titans* the movie.

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. For example from Figure 1 we have:

<"*Clash of Titans*", *directed-by*, *"Louis Leterrier"*>
<"*Clash of Titans*", *genre*, *"action"*>

where the left-hand side entity (e.g. *Clash of Titans*) is the subject *s*, the right-hand side entity (e.g., *Loius Leterrier*) is the object *o*, and the *r* is the relationship linking them (e.g., *directed by*). 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*>).

• **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 2. 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 (Craswell and Szummer 2007). 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 1).

Enriching Word Embeddings

In the following, we present two efficient and simple methods to learn the long range relations based upon the word2vec by encoding the entity relation information.

I. Context and Entity Constrained Model - CECM

Our 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

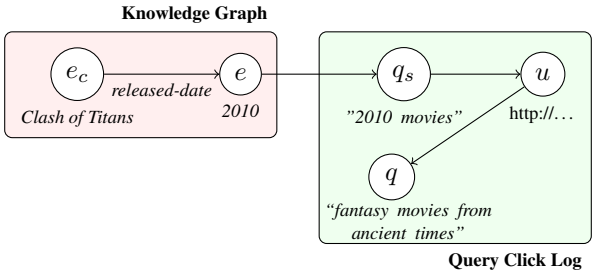

Figure 2: Query mining starting from seed knowledge base entities e_c, then searching the query click logs and later collecting related queries that share the same clicked URL.

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:

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

For instance, for the center word "*zeus*" in the query in Table 1, the window of c=5 includes {*son, of, zeus, in, the, 05szq8z*} and for the center word "*action-advanture*" the window of c=5 includes {*the, 2010, action-advanture, film, </s>, 05szq8z*}. The CECM implicitly encodes the entity relations into the embedding learning algorithm (e.g., the relation between *zeus* and *action-adventure* is reinforced by the entity *05szq8z*). Entity id's are unique to each entity and help to discover for instance that a query with phrase "*... clash of titans ...*" and another with phrase "... the titans" could be pointing to the same entity with id=*05szq8z*.

II. Relation Encoding Model - REM

With the CECM model, the training uses the entities related to the queries (using QET data) as prior information to constrain the local context of words in a given query, ignoring the relational information, which we can leverage from knowledge graph. To *explicitly* encode the relational data into the CECM learning, we propose to use the triplets extracted from freebase and learn entity and relation embeddings. The way we encode the relations is inspired by one of the previous work (Bordes et al. 2013), which is, unlike our work, tries to learn relations between words.

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 (Yu and Dredze 2014). 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

query	freebase entity
q:"*the actor who plays the role of apollo a son of zeus in the 2010 action-adventure film*"	***e***: Clash of the Titans ***t***: freebase.com/m/**05szq8z**

Table 1: Sample $<q, e, t>$ from query-entity-type (QET) dataset.

(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 handside (the relational part) optimizes the relational word embeddings for each relation $R'\in$**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. To be able to apply the above algebraic operation on multi-word entities and relations, before training, we transform each multi-word entity and relation in the QET as well as the triplets data into a compound single word entity by replacing the spaces with '_' (e.g., "*Clash of Titans*" is converted to "*Clash_of_Titans*").

Parameter Estimation. All models (CBOW, RCM, Joint, CECM and REM) use stochastic gradient ascent for learning embeddings and similar parameter values. For each objective we use 15 words as negative samples for each training instance based on their frequency. For pre-training we use standard CBOW with random initialization and use the resulting trained model to initialize the rest of the models.

Experiments

Datasets. To evaluate our approach we use the movie domain dataset from (Hakkani-Tur et al. 2014), 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 com-

piled 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 (Yu and Dredze 2014) 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.

For each embedding model we use 200-dimensional embeddings and output embeddings for analysis. In order to use the word embeddings, which are scalar valued vector representations, as features in semantic tagging task, we use K-means clustering to cluster the n-dimensional word vectors into *K* classes. Specifically, we convert the scalar valued vector representations of words into class-based representation similar to (Bansal and K. Gimpel 2014; Sarikaya et al. 2014). Following (Sarikaya et al. 2014), 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).

Evaluations on Semantic Tagging. We use CRF to build semantic tagging models and add several embedding features from baseline and proposed embedding models as follows:

• 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, RCM, and Joint models respectively.

• {CRF-CECM, CRF-REM, CRF-(CECM->REM)}: Proposed 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.

Table 2 shows that all of our models obtain reductions compared to baselines, with CRF-(CECM->REM) obtaining the largest reductions. The explicit relational embeddings from Freebase can boost the implicit embeddings and together they are better suited for semantic tagging tasks.

	Model	F-Score on Test
Baselines	CRF-1	86.15
	CRF-2	86.09
Baselines with Embeddings	CRF-CBOW	86.61
	CRF-RTM	87.58
	CRF-Joint	87.65
New with Enriched Embeddings	**CRF-CECM**	**88.58**
	CRF-REM	**88.12**
	CRF-(CECM->REM)	**88.70**

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

Conclusion

We have presented new simple, yet effective approaches to learn domain specific word embeddings. We learn the embeddings that together with specific relations they are predictive of words that they have functional relations with. The enriched embeddings provide rich semantic information to the semantic tagging tasks, proving information about the long term relations removing the need for modifying the sequence learning algorithm or collecting large amounts of manually annotated data for training.

References

Bansal, M., and K. Gimpel, a. K. L. 2014. Tailoring continuous word representations for dependency parsing. In *Proc. of ACL*.

Bordes, A.; Usunier, N.; Garca-Durn, A.; Weston, J.; and Yakhnenko, O. 2013. Irreflexive and hierarchical relations as translations. *CoRR* –1–1.

Craswell, N., and Szummer, M. 2007. Random walks on the click graph. *Proc. of ACM SIGIR*.

Daume-III, H.; Langford, J.; and Marcu, D. 2009. Search-based structured prediction. *Proc. of Machine Learning*.

Fellbaum, C. 1999. Wordnet. *Proc. of Wiley Online Library*.

Finkel, J.; Manning, C.; and Ng, A. 2006. Solving the problem of cascading errors: Approximate bayesian inference for lingusitic annotation pipelines. *Proc. of EMNLP*.

Hakkani-Tur, D.; Celikyilmaz, A.; Heck, L.; and Tur, G. 2014. Probabilistic enrichment of knowledge graph entities for relation detection in conversational understanding. *Proc. of Interspeech*.

Lafferty, J.; McCallum, A.; and Pereira, F. 2011. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. of ICML*.

Lebret, R.; Legrand, J.; and Collobert, R. 2013. Is deep learning really necessary for word embeddings ? *Proc. of NIPS*.

Liang, P.; Daume-III, H.; and Klein, D. 2008. Structure compilation: trading structure for features. *Proc. of ICML*.

Martens, J., and Sutskever, I. 2011. Solving the problem of cascading errors: Approximate bayesian inference for lingusitic annotation pipelines. *Proc. of ICML*.

Mikolov, T.; I. Sutskever, K. C.; Corrado, G.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. *Proc. of NIPS*.

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

Tang, D.; Wei, F.; Yang, N.; Zhou, M.; Liu, T.; and Qin, B. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. *Proc. of ACL*.

Weston, J.; Bordes, A.; Yakhnenko, O.; and Usunier, N. 2013. Connecting language and knowledge bases with embedding models for relation extraction. *Proc. of Computation and Language*.

Yu, M., and Dredze, M. 2014. Improving lexical embeddings with semantic knowledge. *Proc. of ACL*.