Supervised Methods for Ranking Relations in Web Search

Sumit Asthana
Dept. of Computer Sc. and Engg.
IIT Patna, Patna
asthana.sumit23@gmail.com

Asif Ekbal
Dept. of Computer Sc. and Engg.
IIT Patna, Patna
asif@iitp.ac.in

Abstract

In this paper we propose an efficient technique for ranking triples of knowledge base using information of full text. We devise supervised machine learning algorithms to compute the relevance scores for item-property pairs where an item can have more than one value.Such a score measures the degree to which an entity belongs to a type, and this plays an important role in ranking the search results. The problem is, in itself, new and not explored so much in the literature, possibly because of the heterogeneous behaviors of both semantic knowledge base and full-text articles. The classifiers exploit statistical features computed from the Wikipedia articles and the semantic information obtained from the word embedding concepts. We develop models based on traditional supervised models like Suport Vector Machine (SVM) and Random Forest (RF); and then using deep Convolution Neural Network (CNN). We perform experiments as provided by WSDM cup 2017, which provides about 1k human judgments of person-profession pairs. Evaluation shows that machine learning based approaches produce encouraging performance with the highest accuracy of 71%. The contributions of the current work are two-fold, *viz.* we focus on a problem that has not been explored much, and show the usage of powerful word-embedding features that produce promising results.

1 Introduction

Most of the prior works in information retrieval (IR) focuses on retrieving information either using semantic knowledge base or text. In the present day, Information Retrieval (IR) often involves both knowledge base as well as full text search. One cannot succeed in retrieving semantic information with the other.Knowledge base is good at returning precise information, whereas full-text has the benefit of a wide information coverage, for example, Wikipedia articles. Therefore, it is imperative that search uses information from both of the above and tries to find a best approximation.

In our current work we discuss the problem to rank, not entities from a full text search but triples from knowledge bases with the same subject and predicate properties. Let us consider all the professions of a particular person, for example of Arnold Schwarzenegger: *Actor, Athlete, Bodybuilder, Businessperson, Entrepreneur, Film Producer, Investor, Politician, Television Director, Writer*. All of them follow: **”Arnold Schwarzenegger—profession—ProfessionName”**, but some of these are more relevant and prominent whereas others are less. Hence it would be good to come up with a metric to segregate the most-relevant ones’ from the less-relevant ones’. The concept of relevance in itself is ambiguous. So here we take the basis as the *amount of information in the Wikipedia article of the entity.*

This type of relevance plays an important role in improving search engines as well as knowledge bases upon which several question-answering systems are being built. For example, all three tasks from the TREC 2011 Entity Track Balog et al. (2011) ask for the lists of entities of a particular type. It is to be noted that ranking of triples using both semantic knowledge base and fulltext articles is not explored at the required level. In order to solve this problem we at first propose models based on supervised machine learning algorithms, namely Support Vector Machine (SVM)

S Bandyopadhyay, D S Sharma and R Sangal. Proc. of the 14th Intl. Conference on Natural Language Processing, pages 227–235,
Kolkata, India. December 2017. ©2016 NLP Association of India (NLPAI)

and Random Forest (RF). Therafter, we develop model based on deep Convolutional Neural Network (CNN).

1.1 Related Works

As already mentioned there have not been required number of attempts for ranking triples. The task has been taken up in Bast et al. (2015) with an unsupervised approach. In Cedeño and Candan (2011), authors have proposed an extension to Resource Description Frameowork (RDF) and they called it as Ranked RDF. A ranking model is proposed in Elbassuoni et al. (2009) for SPARQL queries with possible text extensions based on language models. The technique proposed in Dividino et al. (2012) discusses how to combine several kinds of scores associated with triples into a meaningful ranking. In all these frameworks, scores that are similar to our triple scores are assumed to be given.

We start with the approach given in Bast et al. (2015) and come up with new additional features and methods over the existing one. The key contributions of our current work are as follows: (i). we propose supervised machine learning models for triple ranking that exploits both semantic knowledge base and full text information. This is relatively a new direction of research; and (ii). utilizing word embedding information obtained from the Wikipedia knowledge along with the statistical features. Evaluation of the models on WSDM datasets[1]show encouraging performance. (iii) Through the on-going experiments with deep learning based approaches we show that deep CNN can yield promising results for this type of problem.

2 Problem Description and Dataset

The problem that we tackle is related to ranking the relevance of person-profession pairs based on the information present in Wikipedia. This is an example of a non-functional relation between an entity and an abstract group. We have a set of person names and their associated professions from FreebaseBollacker et al. (2008). The goal is to predict a score for each person-profession relation between 0-7, with 7 being the most relevant. A typical set of training examples is:

Wolfgang Amadeus Mozart	*Composer*	*7*
Wolfgang Amadeus Mozart	*Pianist*	*5*
Wolfgang Amadeus Mozart	*Violinist*	*2*

Table 1: Dataset description

Filename	Description
professions	the 200 different professions from professions.kb
professions.kb	all professions for a set of 343,329 persons
profession.train	relevance scores for 515 tuples (pertaining to 134 persons) from profession.kb
persons	385,426 different person names from the two .kb files and their Freebase ids
wiki-sentences[2]	33,159,353 sentences from Wikipedia with annotations of these 385,426 persons
profession.test	relevance scores for 513 tuples (pertaining to 134 persons) from profession.kb

We use the dataset of WSDM cup-2017 triple[1] scoring task, which provides a training and test set comprising of 1,225 person-profession pairs. Details are shown in Table 1. The labels had been obtained via crowd-sourcing wherein 7 independent judges rated each profession for a person as relevant or non-relevant. The scores of these 7 judges were then added to form the composite score described above.

The training sets (the .train files provided above) contain only tuples from the respective .kb files. The person names are exactly the names used by the English Wikipedia. That is, http://en.wikipedia.org/wiki/PersonName takes you to the respective Wikipedia page. For each of the names in persons, there are sentences in wiki-sentences (68,662 sentences for the most frequently mentioned person, 3 sentences for the least frequently mentioned person).

3 Machine Learning based Approach

In this section we describe our proposed approach which starts with defining the problem and then the specific components on word vector generation, feature extraction, query expansion etc.

3.1 Word Vectors

Word embedding (also known as distributed word representations) persuade a real-valued latent semantic or syntactic vector for each word from a

[1] http://www.wsdm-cup-2017.org/triple-scoring.html

large unlabeled corpus by using continuous space language models. Better word representation can be obtained if we have a large amount of training data as the obtained real-valued vectors of words become more representative. We use the popular *word2vec*[3] tool proposed by Mikolov et al. Mikolov et al. (2013a; Mikolov et al. (2013b) to extract the vector representations of words. Owing to its simpler architecture which reduces the computational complexity, this technique can be used for large corpus. We train Word2Vec tool on the **"wiki-sentences"** corpus. The corpus was first preprocessed by removing all numerals, special symbols, and converting to lowercase. The Word2Vec tool was then trained with *feature size of 400, window size of 8, Continuous Bag-of-Word (BoW) model and min count of 15.*

For each profession and person, we generate the word vectors and concatenate to the respective feature vectors of the instances.

3.2 Query Expansion

We treat every given profession word as a topic and apply the query expansion techniques Bast et al. (2015) to expand the profession to a set of 10 most relevant words related to the profession. For example, the profession *Architect* when expanded yields the following set: *architect,design,building,designed,architectural, buildings,church,built,house*.

3.2.1 Logistic Regression

We learn a Logistic regression (LR) classifier for each profession. The positive instances of the classifier denote the Wikipedia articles of persons who only had that profession as mentions and negative samples correspond to the persons who never had that profession as mentions. We obtain this information from Freebase. The LR classifier is trained with the term frequency matrices of the Wikipedia articles of person.We trained one LR classifier per profession giving us a total of 200 LR classifiers, one for each profession. Each such classifier was trained using positive and negative instances created from the Wikipedia articles. The positive instance articles would be articles of people who only had that profession as mentions. The negative articles are articles of people who did not have that profession mention at all. **Profession mention** for both positive and negative instances came from the **persons** file described above which has all possible valid person-profession pairs for the people in the dataset. Thus, the LR classifier for a profession was trained to learn the distinction between a set of articles segregated on the basis of presence/absence of that profession. As a result, the entity of interest out of this training would be the weights the LR classifiers assigned to each word(features).

Looking at the top scoring word (features), it was clear that they were words somewhat distinguishing the positive and negative instances. The LR parameters were tuned using grid search. We only use the classifier if it has an accuracy of more than equal to 80%. We provide link to our query expansion results that are present on github[4].

For other cases, where the LR classifier failed to segregate instances with sufficient accuracy on account of lack of enough data, the method described below was resorted to.

3.2.2 Using word vectors:

Out of 200 professions, about 40 of them (e.g. entertainer) do not have sufficient training data which could lead to a decent accuracy. For such instances, firstly word embedding vectors are created and then top 10 most similar words are retrieved based on cosine similarities. The word embeddings were trained as described above on the wiki-sentences using the gensim toolkit[5]. We use the most_similar function provided by the word2vec model which takes a word and returns the vectors(and associated words) closest to the given word in cosine similarity.

3.3 Features

After query expansion, we extract the following features for each *Person-Profession-Rank* triple. Features are extracted on the Wikipedia articles of the person. Refer to 3.3 for a graphical overview.

1. Word count on full text(*wcFull*) - This feature denotes the count of indicators (most similar words to a profession) and all profession words are present on Wikipedia article of a person.

2. Word count in opening text(*wcOpen*) -This feature corresponds to the count of indicators

[3] https://code.google.com/p/word2vec/

[4] https://github.com/codez266/turnip/blob/master/indicators-pro

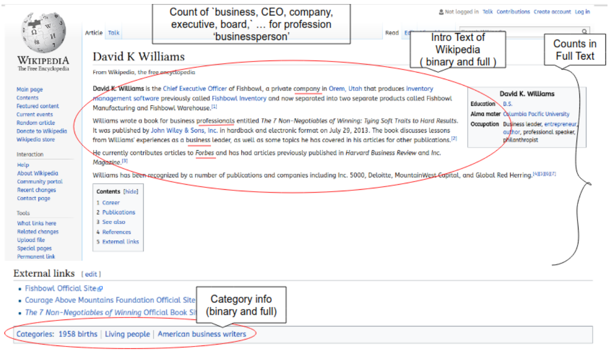
[5] https://radimrehurek.com/gensim/models/word2vec.html

Table 2: Scores for methods without word vectors(accuracy represents exact matches)

Method	Accuracy(δ=0)	Average Score Difference	Kendall's Tau
Counting(Baseline)	0.68	1.92	0.42
SVM	0.69	1.86	0.37
Random Forest	0.71	1.80	0.34

Table 3: Scores for methods(accuracy represents exact matches)

Method	Accuracy(δ=0)	Average Score Difference	Kendall's Tau
Counting(Baseline)	0.69	1.90	0.39
SVM	0.70	1.86	0.35
Random Forest	0.71	1.78	0.33



David K Williams

From Wikipedia, the free encyclopedia

David K. Williams is the Chief Executive Officer of Fishbowl, a private company in Orem, Utah that produces inventory management software previously called Fishbowl Inventory and now separated into two separate products called Fishbowl Manufacturing and Fishbowl Warehouse.[1]

Williams wrote a book for business professionals entitled *The 7 Non-Negotiables of Winning: Tying Soft Traits to Hard Results*. It was published by John Wiley & Sons, Inc. in hardback and electronic format on July 29, 2013. The book discusses lessons from Williams' experiences as a business leader, as well as some topics he has covered in his articles for other publications.[2]

He currently contributes articles to *Forbes* and has had articles previously published in *Harvard Business Review* and *Inc. Magazine*.[3]

Williams has been recognized by a number of publications and companies including Inc. 5000, Deloitte, MountainWest Capital, and Global Red Herring.[4][5][6][7]

David K. Williams	
Education	B.S.
Alma mater	Columbia Pacific University
Occupation	Business leader, entrepreneur, author, professional speaker, philanthropist

External links [edit]

- Fishbowl Official Site
- Courage Above Mountains Foundation Official Site
- *The 7 Non-Negotiables of Winning* Official Book Si

Categories: 1958 births | Living people | American business writers

Figure 1: Features on Wikipedia article

and all profession words present in the introduction text of Wikipedia article of person.

3. Word count in Category(*catCount*) -This indicates the count of all profession words in the category section of Wikipedia article.

4. Binary presence in full text(*catBin*) - This denotes the presence or absence of profession words in the category section of an article.

5. Presence in opening text(binary)(*bOpen*) - This feature denotes the presence or absence of all profession words in opening text of Wikipedia article of person.

6. (*wVec*) - This feature is defined based on the word embedding vectors as defined earlier. It is obtained by concatenating the word vectors of a person name and profession name. E.g. for *Wolfgang Amadeus Mozart Composer 7* we concatenate vectors of **"Wolfgang Amadeus Mozart"** and **"Composer"** to form an 800 dimension vector and add it to the existing vector of other features as described in this section.

Finally, we input a feature vector of dimension 805 to the classifiers.

3.4 Justification for using the additional word vectors as features

As word2vec(Mikolov et al., 2013b) mentions learning information about various features of words with respect to their context, this information in encoded in the dimensions of the vector. We intend to use this dimensional information as an input to the classifier so check if the contextual information contains some signal to distinguish professions or not.

3.5 Classifiers

We develop models using three classifiers. We use the scikit-learn library for implementation of these classification models. The grid-search[6] module was used to optimize the above set of parameters and get the best performing set.

1. Scores based on just the normalized raw counts of words for professions associated with a person. This method formed the baseline and as such did not use any classifier. The scoring was done based on normalizing raw counts across person-profession pairs for the same person.

2. SVM classifier which is developed with the above set of features(Sec 3.3).

3. Random Forest classifier developed with the above set of features(Sec 3.3).

For the last two cases, the instances with score 0-3 are mapped to label 0 and instances with score 4-7 are mapped to 1. During testing, binary output of

[6] http://scikit-learn.org

the classifier is projected in a similar manner to get the final scores. The label prediction 0 is mapped to 0-3 using the normalized raw counts as per the first approach. The label prediction 1 is mapped to 4-7 using the same approach. An example of using normalized raw counts to generate scores: For a person X, consider the professions with raw counts of associated words:

- Actor - 20
- Director - 10
- screenwriter - 7

As per the information, actor would get a rating of 6-7, director would be scored as 3-4 and screenwriter roughly 2-3.

The **reason** for adopting this hybrid approach was that final rankings had to be from 0-7 which reflected the degree of belongingness of profession to the person. However, this is a very fine-grained scoring for a classifier and a ranking of 2/3 or 4/5 isn't much different. If we had used seven different labels, the classifier would have tried to draw a fine decision boundary across all seven classes, which isn't feasible. Therefore, we thought it best to use a classifier to segregate between relevant and non-relevant, and then adjust in a post-processing normalization step to generate scores with the dataset requirements(i.e between 0-7). This also has the benefit of being close to how the users rated the person-profession pairs. (Bast et al., 2015) mentions that each user chose between relevant and non-relevant when presented with the person-profession pair during the training step.

We now move on to provide details about the classification.

1. ***Counting Approach: Baseline***: The baseline model that we define is based on counting profession word and its indicators in the article of the person whom we have to rate. Only profession word is not always indicative of the actual person-profession relation. For example, let us consider, *"Jolie made her screen debut as a child alongside her father"*. Here, **"screen"** or **"debut"** somewhat convey an acting profession. Hence, this observation necessitates the need for finding more relevant words (i.e. indicator words) related to a profession, which we also include in the counting alongwith the main profession word. This is the simplest approach which involves counting the profession and its indicator words and normalizing them linearly, with the greatest of them achieving the score 7 and accordingly.

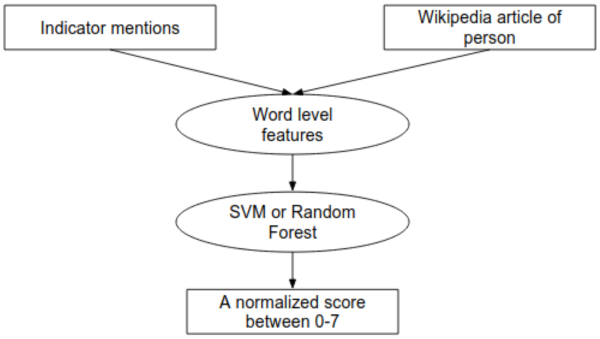


Figure 2: Pipeline for Classification

2. ***SVM based Approach***: With a combination of the features as described above, SVMs (Joachims, 2002) are trained to learn the relevance. Grid search is used to the tune given parameters:(Best - *Kernel: rbf, C: 1, Gamma: 1*)
   - Kernel: *rbf, linear*
   - C(Penalty Parameter): *0.01,0.1,1,10,100*
   - Gamma(Kernel Coefficient): *0.001,0.01,0.1,1,10*
3. ***Random Forest based Approach***: Similar to SVMs, we use the same set of features to learn Random Forest (Breiman, 2001). The grid search parameters are set as:(Best - *max_features: sqrt, n_estimators: 10, min_samples_split: 0.05*)
   - oob_score: *True*
   - max_features: *sqrt, log2*
   - n_estimators: *10, 100*
   - min_samples_split: *0.05, 0.10, 0.15, 0.20*

3.6 End to end pipeline

The relevance scoring mechanism consists of the following stages: Fig. 2 shows the basic way of training the classifier.

1. Indicator words generation for professions for which enough data is available in the form of articles of people in that profession. This

step uses learning an LR classifier per profession as described earlier. In parallel, we use word2vec to generate indicator words for the less prominent professions.

2. Each training instance is a person-profession-rank triple and the test instance a person-profession pair. We use the Wikipedia article of the person and the set of 10-15 indicator words so generated along with the original profession words to generate the feature values on the article. We get a feature vector of length five from this. We append the additional 800 dimensional vectors generated through word embedding (by Word2vec tool) for the profession and person in each instance. This produces a resulting vector of 805-dimension.

3. This vector is fed to the classifier discussed above, which was trained to do a binary classification of relevant/non-relevant.

4. These binary classification labels from the above classifier were then scaled to the values between 0-7 (discussed in the introduction of classifiers section) to conform to the output standard for analysis.

4 Experiments and Analysis

4.1 Evaluation

We use three metrics to measure the efficiency of the baseline and the proposed models.

1. Accuracy - The percentage of person-profession triples that matched.

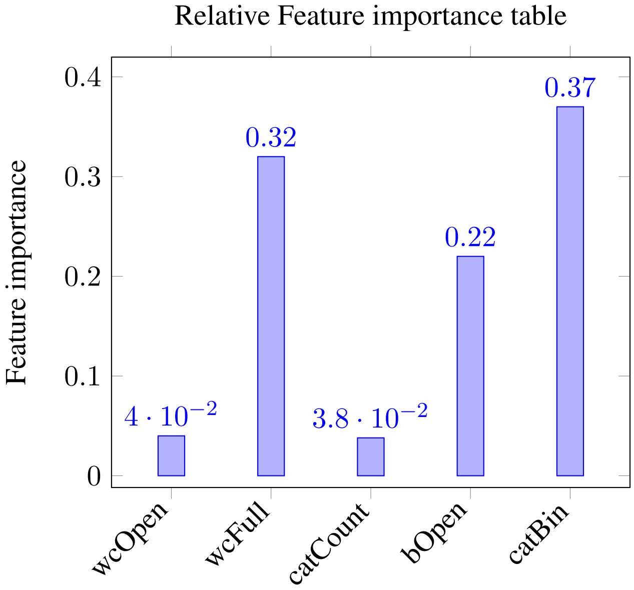
2. Kendall's tau - $\tau_p = 1/Z(n_d + p.n_t)$ where n_d is the number of discordant (inverted) pairs, n_t is the number of pairs that are tied in the gold standard but not in the predicted ranking or vice versa, p is a penalization factor for these pairs which we set to 0.5, and the normalization factor Z (the number of ordered pairs plus p times the number of tied pairs in the gold standard). This is to account for the tied rankings in the gold standardFagin et al. (2004).

3. Average Score Difference - Average of the difference of scores between gold mention and predictions.

4.2 Scores and Best Features

We perform 10-fold cross validation on the training data for optimizing the model and evaluate on the the test set. **Table 2** shows the scores for SVM and Random Forest along with the baseline. It shows that random forest based model performs slightly better than SVM. However, both of these approaches perform better compared to the baseline model.The 800-dimension concatenated word vector of person and profession did not provide generalization as we expected. A possible reason could be the insufficient size of **wiki-sentences** with only 33000k sentences, which were used for training of Word2Vec tool. However it is to be noted that that word embedding vectors were more useful for generating the indicator words of a profession.

We measure the importance of each feature and its effect (except word vectors). Importance to these features is extracted using **feature_importances_** data structure provided by scikit-learn after training them.



Clearly, **wcOpen**(word count in opening text) and **catCount**(word count in category) do not have convincing roles. A possible reason for this might be that:

1. *Opening text* of Wikipedia and *category* do not often enumerate all the professions.

2. Their relative counts in a small paragraph are not sufficient.
The latter observation is backed by the fact that binary word presence in the first para

(**bOpen**) is a good feature where we only account for the presence or absence of a profession word.

3. As expected, the full-text search of profession words along-with its indicators is a significant feature.

4. The most important thing is the importance of the binary feature of profession word in **category catBin**, which shows that Wikipedia categories are the reflection of subject matter in a comprehensive manner.

4.3 Error Analysis

We perform a thorough analysis to understand the shortcomings that still need to be tackled:

1. *Popularity*: The way human ranked the persons is not very well defined and is hugely affected by popularity. Popular personalities get ranks for professions based on a lot more prior knowledge than just a Wikipedia article. Unpopular personalities are assigned ranks based on what is directly visible in the initial glance of the Wikipedia article, in most cases.

2. *Amount of information*: The ranks were also affected by the amount of information present in the Wikipedia article of a person. For example, **Ba. U** has been rated as **2 for politician** and **7 for Lawyer**, although he has been actively involved in politics, having been the president two times. The only issue is that his Wikipedia article is too short to provide a substantial information.

3. *Drawback of a linear relationship based on word count*: It is clear that the underlying idea of indicator word count is not so much useful. Often for very long articles the count seems to lose its meaning. For example, **Napoleon** has been rated as **7** for both **politician** and **military officer**, but given his long description of military campaigns, military officer seems to outweigh politician during prediction.

4. *Experiments with Word Vectors* The word vectors were trained on the wiki-sentences to get the context from the Wikipedia mentions. The use of word embedding vectors improves the accuracy to some extent, and greatly helps in deriving more contextual information for 25% profession words for which we had very less person mentions from Freebase. This shows that they can be used in places where we have insufficient information in semantic space to derive context.

4.4 Comparison with previous work

We'd like to mention that the previous work (Bast et al., 2015) achieved an overall accuracy of 63% with their method **MLE combined**. We achieved an overall accuracy of about 70%. However, they do better on the average score difference front, getting 1.57 as best with the **Count Combined method**. We report the best average score difference as 1.78 with Random Forests and word vector features. Kendall's tau is 0.22 for them with **MLE combined** whereas its 0.33 for us with random forests. However, we mention that our training data was significantly less than (Bast et al., 2015) because they used the entire Wikipedia for training whereas we only used wiki-sentences for word vector generation and individual Wikipedia articles of persons for feature generation per instance.

4.5 Relation to web-search

We mention **Web search** in the title because getting relevance scores for several entity relation pairs with different predicates but same entity can help to show only the most significant one's in cases where only one or two results are required. This is especially useful for filtering in today's world of search where search engines also take information from knowledge bases like Wikidata[7].

5 Deep Learning based Approach

We exploit deep learning algorithm being motivated from the fact that it does not require any feature engineering. For our deep learning based approach, we develop a model based on CNN. CNNs are able to convolve over the entire text just like on images and are able to extract features from the text efficientlyKim (2014). We present the results achieved till now using CNN to identify single relation entities, e.g people with only a single profession. Though not complete, this step is important as it can be extended to the multi-profession case. The final ranking generation for

[7] https://www.wikidata.org/

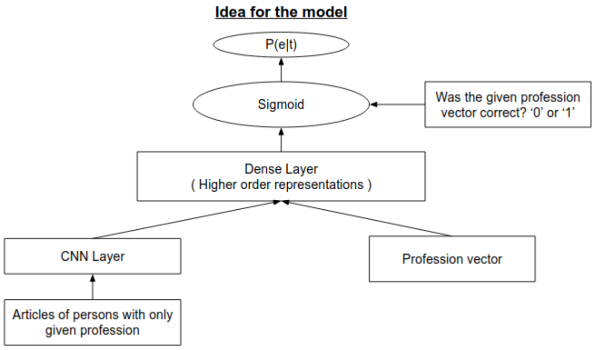

Idea for the model
P(e|t)
Sigmoid
Was the given profession vector correct? '0' or '1'
Dense Layer
(Higher order representations)
CNN Layer
Profession vector
Articles of persons with only given profession

Figure 3: Intuition for CNN

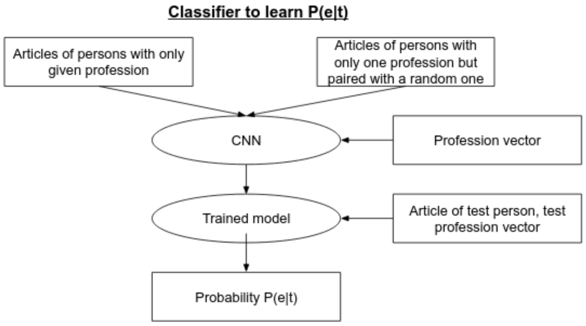

Classifier to learn P(e|t)
Articles of persons with only given profession
Articles of persons with only one profession but paired with a random one
CNN
Profession vector
Trained model
Article of test person, test profession vector
Probability P(e|t)

Figure 4: Deep network classifier for profession

multi-relation entities using CNNs is left as a followup of this work or can be taken up anywhere else.

Unlike the previous model, which had a different classifier for identifying indicator words for each profession, here we use a single deep network classifier to find $P(e|t)$ where **'e'** is the person entity and **'t'** is the type(profession), i.e, we find the degree of belongingness of the entity to the type.

Fig. 4 shows the basic way of training the classifier which is quite similar to one described in the previous approach. We use both Convolution Neural Network (CNN) and Long Short Term Memory (LSTM), but we found better results with CNN and hence report the results using only this classifier.

The only difference is that we now use a single classifier to classify across all person-profession pairs and we now also include profession information as a word vector of profession. The underlying idea is to learn *common high level representations that make a person and a profession similar*.

Fig. 4 shows the basic way of training the classifier.

Fig. 3 shows the basic overview of the CNN model.

Table 4: CNN statistics

Samples	22638
Hidden Layers	2
Hidden Neurons	512, 64
Embedding Dimension	300
Precision	82%

Parameters to the CNN classifier:

- Google-News word vectors[8] of dimension 300.
- 50 filters of size 3 each and one convolutional layer.
- GlobalMaxPooling and AveragePooling but found that MaxPooling performed better in all cases.
- Two dense layers after convolution, which were formed after **merging convolved Wikipedia article of person and profession vector** Fig. 3.
- The first dense layer has **512** neurons and second one has **64** neurons.

Table 4 shows statistics using CNN classifier which was implemented using keras[9]. For efficient scoring, we considered only first two thousand letters in the Wikipedia article of each person.

It was found that CNN based classifier performs very well while classifying single-profession entities as correct/incorrect pair, but when extended to multi-profession entities it was somewhat not able to distinguish the more relevant professions from the less relevant one's. We attribute this to the noise introduced by several professions in the Wikipedia article of the person and leave this as an interesting task to explore as a follow-up of this work, or elsewhere.

6 Conclusion

In this paper we have proposed supervised machine learning based solutions for handling triple ranking in a mixed domain of knowledge base and

[8] code.google.com/p/word2vec/

[9] keras.io

full-text. We have explored two supervised classifiers with handcrafted features extracted on English Wikipedia along with word embeddings to learn the rankings. Some of the features work well to learn the rankings but more can be explored. Moreover, by using CNN as a classifier to learn representations of person-profession entities, we have shown that deep learning can be applied to this domain and provide and interesting alternative method to explore further.

References

Krisztian Balog, Pavel Serdyukov, and Arjen P. de Vries. 2011. Overview of the TREC 2011 entity track. In *Proceedings of The Twentieth Text REtrieval Conference, TREC 2011, Gaithersburg, Maryland, USA, November 15-18, 2011*.

Hannah Bast, Björn Buchhold, and Elmar Haussmann. 2015. Relevance scores for triples from type-like relations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 243–252. ACM.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*, pages 1247–1250. ACM.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Juan P. Cedeño and K. Selçuk Candan. 2011. R^{2}df framework for ranked path queries over weighted RDF graphs. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics, WIMS 2011, Sogndal, Norway, May 25 - 27, 2011*, page 40.

Renata Queiroz Dividino, Gerd Gröner, Stefan Scheglmann, and Matthias Thimm. 2012. Ranking RDF with provenance via preference aggregation. In *EKAW*, volume 7603 of *Lecture Notes in Computer Science*, pages 154–163. Springer.

Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum. 2009. Language-model-based ranking for queries on rdf-graphs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 977–986.

Ronald Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, and Erik Vee. 2004. Comparing and aggregating rankings with ties. In *PODS*, pages 47–58. ACM.

T. Joachims. 2002. *Learning to Classify Text Using Support Vector Machines – Methods, Theory, and Algorithms*. Kluwer/Springer.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751. ACL.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.