ExSim at eHealth-KD Challenge 2020

Combining NLP and Word Embeddings for Entity Recognition

Zainab Hamzah Almugbel[1,2][0000−0003−4570−7088]

[1] University of Leeds, Leeds, UK
[2] Imam Abdulrahman Bin Faisal University, Dammam, SA
sczha@leeds.ac.uk

Abstract. This paper describes the system submitted to the eHealth-KD Challenge 2020-Task A: entity recognition. The system utilizes a supervised learning methodology to recognize entities within Spanish texts; namely, it applies NLP and word2vec techniques to create a unique labeled dictionary of entities in the training set. These labels are propagated into new entities that are found in the testing set via semantic similarity measurement. The simplicity of our system shows low performance with F1=0.32, precision= 0.29, and recall=0.34. Finally, the system is discussed from different aspects: challenges, earlier attempts, current system's characteristics, and possible future work.

Keywords: entity recognition · NLP · word2vec.

1. Introduction

Entity recognition (ER) plays vital role in analyzing unstructured texts. ER is utilized in several domains for different applications, such as information retrieval and recommendation systems; specifically, ER is widely utilized to recognize health related entities in healthcare domain, e.g., diseases and drugs names [1]. Because ER is widely applied, research propose several approaches to deal with this problem. These approaches can be categorized into two sections. The first section is called the supervised learning approaches. They require a training set, such as applying classifiers on annotated words [2]. The second section is called the unsupervised learning approaches. They require statistical models, such as classifying entities based on TF-IDF schema [3].

The eHealth-KD 2020 Task A [4] aims at recognizing entities in unstructured Spanish health sentences that are taken from Medline. This paper proposes a supervised learning approach to tackle the ER problem. This means training a given annotated text (training set) to recognize and label entities in new text (testing set). Our approach is motivated by the theory that defines the language

as a bag of labelled entities [5] and the research that states the importance of measuring the similarity of meanings (semantic similarity) among these entities in both linguistics field and classification task [6].

In the research [6], semantic similarity is utilized to learn how to identify verbs of different tenses or the capitals of countries. This means that if the label of an entity is given, it can be assigned to new similar entity. For instance, if the country's label is "noun" in the capitals of countries example, this label can be also assigned into the capital's label. Hence, semantic similarity can identify entities that belong to same label. However, one obvious drawback of this approach is that it requires large amount of training data to minimize the possibility of "word out of vocabulary" error.

Based on that, ER problem can be treated as a classification problem. The classes are the entities' labels that are taken from the training set; they are assigned to one of the following values: concept, action, predicate, and reference. The semantic similarity measurement can be employed as the classifier. It labels the new entities according to their similarity measurement to the exist entities. Each new entity is labelled by the label of the entity with the highest similarity score.

Thus, the system's inputs are: 1. the annotated texts that include the entities and their labels, and 2. the target unstructured text that includes unlabelled entities. The system's output is the predicated labels of the identified entities that exist within the target unstructured text. Each entity has only one label. In order to identify entities and predict their labels, the system applies NLP [7] for text cleaning, and word embeddings [8] for representing entities as numerical vectors. Then, it calculates the cosine similarity among these vectors to measure their semantic similarity.

The rest of the paper presents the system description in Section 2, followed by the results in Section 3. Then, the discussion is disclosed in section 4. Finally, the conclusion is stated in section 5.

2. System Description

This section mainly explains the system. We propose a simple idea for identifying the medical entities. It is about using NLP and word2vec to create a dictionary that contains distinct entities with their labels from the training set based on similarity measurement. After Algorithm 1 creates this dictionary, it is utilized to annotate the entities of the sentences in the testing set. More detail about the system is discussed next. It is organized in three subsections for clarification purposes: algorithm, text cleaning and word embeddings.

2.1. The Algorithm

In this part, the algorithm is presented, and its main parameters and variables are explained.

Algorithm 1: Entity Recognition algorithm

Result: a list of entities with their predicted labels

keyphrases=[[K_1, L_1],[K_2, L_2],...,[K_m, L_m]] ;
TrainingList= [[se_{11},se_{12},se_{13},...][se_{21},se_{22},...]....[...,se_{ij}]] ;
model=word2vec(TrainingList,...);
TestedTokens=[[]];
for *each K,L in keyphrases* **do**
  for *each sentence s in testing set* **do**
    if *exact match of K in s* **then**
      testedTokens.append([K, 1, L])
    end
    preprocess(s);
    for *each token t in s* **do**
      score=model.similarity(t,K);
      if *score>0* **then**
        if *t not in testedTokens* **then**
          testedTokens.append([t, score, L]);
          else if *current score > stored score* **then**
            testedTokens.update([t, current score, current L])
      end
    end
  end
end

Algorithm 1 consists of three lists that require clarification:

- `keyphrases` is a nested list that contains each entity and its annotated label. For instance, K_m is the entity and L_m is its label, where the label is one of four possible values (concept, predicate, reference, action).
- `TrainingList` is also a nested list that contains the entities of sentences. se_{ij} is the entity **j** of the sentence **i**; where **i** is for the sentences and **j** is for the entities within a sentence, e.g. se_{11} is the first entity in the first sentence.
- `TestedTokens` list contains a list of three elements $\mathbf{[}t_n, Score_n, L_n\mathbf{]}$. Each list's element is unique because each entity t_n must have at most one label L_n.
  1. The first element t_n is an entity that is taken from the testing set.
  2. The second element $Score_n$ is the similarity measurement score between t_n and the entity K.
  3. The third element L_n is the predicted label for t_n.

Both **pre-processing** and **model** are explained in the next subsections.

2.2. Text Cleaning

In this system, text cleaning is applied only on the testing set. This is because the **model** is directly trained on the **TrainingList** that contains the annotated

entities of the sentences in lower case letters. The **preprocess** function is applied to the sentences in the testing set. It employs Natural Language Toolkit (NLTK) to clean each sentence by removing punctuation, any entities with parenthesis, and stop words [7]. Then, it splits the sentence into a list of entities in lower case letters.

2.3. Word Embeddings

The system employs Gensim library to apply word2vec [8] model for word embeddings and similarity measurement. First, the model is trained on the list: `TrainingList` with the following Hyper-parameters:

- `size=100`: it is the dimensional size of the vectors that represent the entities.
- `window=5`: it is the maximum distance between the current and predicted entity within a sentence.
- `min_count=5`: it is the minimum considered frequency of entities in the training model.
- `workers=10`: it is the number of threads to be used by the training model
- `skip-gram` model is used in the training model.
- `epochs=20`: it is the number of the iterations over the training set.

Second, the model calculates the cosine similarity between an entity from the training set and an entity from the testing set. A threshold value is not considered because the system assigns the label of the highest similar labelled entity for each entity.

3. Results

This section reports the teams performance in the selected task [4]; specifically, Figure 1 shows the results of our system for both development and testing sets. Figure 2 shows the results for all participating teams in the same task. In general, one can notice that our system failed to identify entities. This might be because the model is limited by the entities of the training set.

4. Discussion

This section illustrates the challenges that we faced to participate in this competition, some earlier attempts to improve the system and the current system characteristics.

First, two of the top most challenges were the language and the computing resources. First, the language barriers could be from two aspects: personal and technical. In the personal side, some translations had to be carried out to understand some entities because we are not familiar with Spanish. In the technical side, when we tried to implement the system from scratch, we had a problem with writing on the output files. The system shows results on the console but

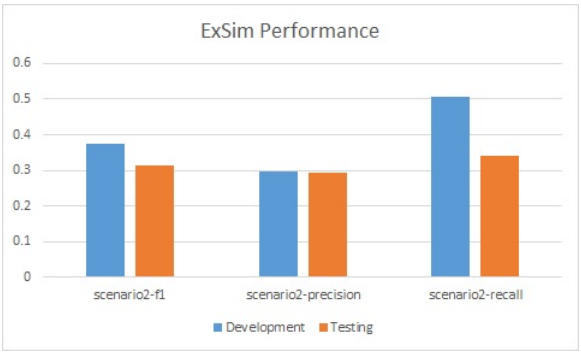

Fig. 1. ExSim Results in Task A

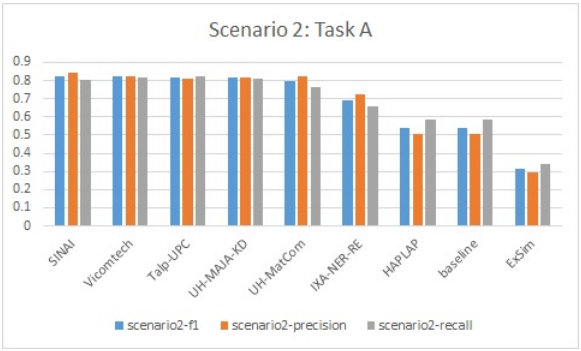

Fig. 2. Final Teams Results in Task A

the output files are empty. After the failure of several attempts to solve this encoding problem, this earlier implementation was neglected, and eHealth-KD 2020's baseline was utilized instead. Second, the system utilized the followings computing resources: i7-6500 CPU 2.50GHz and 8GB RAM. These resources had limitations from two sides: the run time and the memory capacity, as discussed in the second and third points.

Second, the earlier attempts to improve the system include the followings:

- FastText was applied for word embeddings but it produced fake similarity measurement scores for most entities. This caused labelling most entities with one label ,e.g. concept. We modified its hyper-parameters to check how this might influence the similarity scores but the scores were not improved. Therefore, it is replaced with word2vec that gives more reliable scores. The main purpose of using FastText is because it works on characters' level. This could facilitate identifying the entities that share specific number of characters, which in turn improve matching similar entities of different lengths.
- In another earlier attempt, both uni-grams and bi-grams were considered but this was also canceled because it did not improve the system perfor-

mance. The following steps clarify our approach for measuring the semantic similarity among the bi-grams:

1. The bi-grams (Training phrases) with their labels are extracted from the training set.
2. A list of the possible bi-grams (Testing phrases) for each sentence is created from the testing set.
3. The semantic similarity between the entities of training phrases and testing phrases are aggregated; then, it is averaged.
4. The value of the training phrase's label is assigned to the testing phrase' label based on the highest average score.

- Using external trained model [9] was also considered but the system run into "out of memory" error.

Third, the current system characteristics are as follows:

- The last version of the system has not considered POS tagging and word lemmatization.
- The last model's hyper-parameters are set after several modifications:
  - The `min_count` parameter has been tested for the values 1 to 5. The value 5 is chosen. Although this causes skipping some entities, it decreases the run time to 12 hours.
  - Other parameters have been also tested for different values but we have not observed any improvement in the system. These include the following parameters:
    - `size` was tested for the values 60-300.
    - `workers` was tested for the values 5-10.
    - `epochs` was tested for the values 10-20

5. Conclusion

This paper presents the system that is implemented to deal with the entity recognition task. It is based on NLP and word embeddings techniques. Since the system has low performance, it requires improvements from different aspects. First, better computing resources is recommended. This brings many benefits to the system: 1. lowering the system's running time, and 2. enabling the usage of pre-trained models and external resources. Second, POS tagging should be applied to remove none important words. Third, conducting a literature review is encouraged. This supports getting the advantage of high performance previous ER techniques.

6. Acknowledgements

We thank the organizers and the reviewers of the eHealth-KD 2020. Special thanks go to the supervisors Prof. Eric Atwell for his valuable suggestions, and Dr. Mohammad Ammar Alsalka for his helpful comments.

References

1. Unanue, I.J., Borzeshi, E.Z., Piccardi, M.: Recurrent neural networks with specialized word embeddings for health-domain named-entity recognition. Journal of biomedical informatics 76, 102–109 (2017)
2. Florian, R., Ittycheriah, A., Jing, H., Zhang, T.: Named entity recognition through classifier combination. In: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4. pp. 168–171. Association for Computational Linguistics (2003)
3. Zhang, S., Elhadad, N.: Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. Journal of biomedical informatics 46(6), 1088–1098 (2013)
4. Piad-Morffis, A., Gutiérrez, Y., Estevez-Velarde, S., Almeida-Cruz, Y., Muñoz, R., Montoyo, A.: Overview of the ehealth knowledge discovery challenge at iberlef 2020. In: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020) (2020)
5. Burr, V.: An introduction to social constructionism. Routledge (2006)
6. Jatnika, D., Bijaksana, M.A., Suryani, A.A.: Word2vec model analysis for semantic similarities in english words. Procedia Computer Science 157, 160–167 (2019)
7. Bird, S., Klein, E., Loper, E.: Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc." (2009)
8. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010), http://is.muni.cz/publication/884893/en
9. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)