# Self-Attention Gazetteer Embeddings for Named-Entity Recognition

## Abstract

Recent attempts to ingest external knowledge into neural models for named-entity recognition (NER) have exhibited mixed results. In this work, we present GazSelfAttn, a novel gazetteer embedding approach that uses self-attention and match span encoding to build enhanced gazetteer embeddings. In addition, we demonstrate how to build gazetteer resources from the open source Wikidata knowledge base. Evaluations on CoNLL-03 and Ontonotes 5 datasets, show F1 improvements over baseline model from 92.34 to 92.86 and 89.11 to 89.32 respectively, achieving performance comparable to large state-of-the-art models.

## Introduction

Named-entity recognition (NER) is the task of tagging relevant entities such as person, location and organization in unstructured text. Modern NER has been dominated by neural models BIBREF0, BIBREF1 combined with contextual embeddings from language models (LMs) BIBREF2, BIBREF3, BIBREF4. The LMs are pre-trained on large amounts of unlabeled text which allows the NER model to use the syntactic and semantic information captured by the LM embeddings. On the popular benchmark datasets CoNLL-03 BIBREF5 and Ontonotes 5 BIBREF6, neural models with LMs achieved state-of-the-art results without gazetteers features, unlike earlier approaches that heavily relied on them BIBREF7. Gazetteers are lists that contain entities such as cities, countries, and person names. The gazetteers are matched against unstructured text to provide additional features to the model. Data for building gazetteers is available for multiple language from structured data resources such as Wikipedia, DBpedia BIBREF8 and Wikidata BIBREF9.

In this paper, we propose GazSelfAttn, a novel gazetteer embedding approach that uses self-attention and match span encoding to build enhanced gazetteer representation. GazSelfAttn embeddings are concatenated with the input to a LSTM BIBREF10 or CNN BIBREF11 sequence layer and are trained end-to-end with the model. In addition, we show how to extract general gazetteers from the Wikidata, a structured knowledge-base which is part of the Wikipedia project.

Our contributions are the following:

[topsep=1pt, leftmargin=15pt, itemsep=-1pt]

We propose novel gazetteer embeddings that use self-attention combined with match span encoding.

We enhance gazetteer matching with multi-token and single-token matches in the same representation.

We demonstrate how to use Wikidata with entity popularity filtering as a resource for building gazetteers.

GazSelfAttn evaluations on CoNLL-03 and Ontonotes 5 datasets show F$_1$ score improvement over baseline model from 92.34 to 92.86 and from 89.11 to 89.32 respectively. Moreover, we perform ablation experiments to study the contribution of the different model components.

## Related Work

Recently, researchers added gazetteers to neural sequence models. BIBREF12 demonstrated small improvements on large datasets and bigger improvements on small datasets. BIBREF13 proposed to train a gazetteer attentive network to learn name regularities and spans of NER entities. BIBREF14 demonstrated that trained gazetteers scoring models combined with hybrid semi-Markov conditional random field (HSCRF) layer improve overall performance. The HSCRF layer predicts a set of candidate spans that are rescored using a gazetteer classifier model. The HSCRF approach differs from the common approach of including gazetteers as an embedding in the model. Unlike the work of BIBREF14, our GazSelfAttn does not require training a separate gazetteer classifier and the HSCRF layer, thus our approach works with any standard output layer such as conditional random field (CRF) BIBREF15.

BIBREF16 proposed an auto-encoding loss with hand-crafted features, including gazetteers, to improve accuracy. However, they did not find that gazetteer features significantly improve accuracy.

Extracting gazetteers from structure knowledge sources was investigated by BIBREF17 and BIBREF18. They used Wikipedia's instance of relationship as a resource for building gazetteers with classical machine learning models. Compared to Wikidata, the data extracted from Wikipedia is smaller and noisier.

Similar to this paper, BIBREF19 used Wikidata as a gazetteer resource. However, they did not use entity popularity to filter ambiguous entities and their gazetteer model features use simple one-hot encoding.

## Approach ::: Model Architecture

We add GazSelfAttn embeddings to the popular Neural CRF model architecture with ELMo LM embeddings from BIBREF2. Figure FIGREF5 depicts the model, which consists of Glove word embeddings BIBREF20, Char-CNN BIBREF21, BIBREF1, ELMo embeddings, Bi-LSTM, and output CRF layer with BILOU (Beginning Inside Last Outside Unit) labels encoding BIBREF22. Note that, we concatenate the gazetteer embeddings to the Bi-LSTM input.

## Approach ::: Gazetteers

In this section, we address the issue of building a high-quality gazetteer dictionary $M$ that maps entities to types, e.g., $M$[Andy Murray] $\rightarrow $ Person. In this work, we use Wikidata, an open source structured knowledge-base, as the source of gazetteers. Although, Wikidata and DBpedia are similar knowledge bases, we choose Wikidata because, as of 2019, it provides data on around 45 million entities compared to around 5 million in DBpedia.

Wikidata is organized as entities and properties. Entities can be concrete (Boston, NATO, Michael Jordan) and abstract (City, Organization, Person). Properties describe an entity relations. For example, Boston instance_of City and Boston part_of Massachusetts; both instance_of and part_of are properties. Also, each entity is associated with sitelink count which tacks mentions of the entity on Wikimedia website and can be used as proxy for popularity.

To extract gazetteers from Wikidata, we process the official dumps into tuples of entity and type based only on the left and right part of the instance_of triplet, example resulting tuples are Boston $\rightarrow $ City and Massachusetts $\rightarrow $ State. Each entity is associated with a set of aliases, we keep only the aliases that are less than seven tokens long. Example aliases for Boston are “Beantown” and “The Cradle of Liberty”. If there are multiple types per alias, we use the sitelink count to keep the six most popular types. The sitelink filtering is important to reduce the infrequent meanings of an entity in the gazetteer data.

The Wikidata types that we obtain after processing the Wikidata dumps are fine-grained. However, certain NER tasks require coarse-grained types. For instance, CoNLL-03 task has a single Location label that consists of cities, states, countries, and other geographic location. To move from fine-grained to coarse-grained types, we use the Wikidata hierarchical structure induced by the subclass_of property. Examples of subclass_of hierarchies in Wikidata are: City $\rightarrow $ Human Settlement $\rightarrow $ Geographic Location, and Artist $\rightarrow $ Creator $\rightarrow $ Person. We change the types granularity depending on the NER task by traversing up, from fine-grained types to the target coarse-grained types. For instance, we merge the Artist and Painter types to Person, and the River and Mountain types to Location.

## Approach ::: Gazetteer Matching

Gazetteer matching is the process of assigning gazetteer features to sentence tokens. Formally, given a gazetteer dictionary $M$ that maps entities to types, and a sentence $S = (t_1, t_2, ..., t_n)$ with tokens $t_i$, we have to find the $m$ gazetteer types $\lbrace g^1_i, g^2_i,..,g^m_i\rbrace $ and spans $\lbrace s^1_i, s^2_i,..,s^m_i\rbrace $ for every token $t_i$. The set notation $\lbrace $} indicates that multiple $m$ matches are allowed per token. The match span $\lbrace s^j_i\rbrace $ represents positional information which encodes multi-token matches. The match spans are encoded using a BILU (Beginning Inside Last Unit) tags, similar to the BILOU tags that we use to encode the NER labels.

In general, there are two methods for gazetteer matching: multi-token and single-token. Multi-token matching is searching for the longest segments of the sentence that are in $M$. For instance, given $M$[New York] $\rightarrow $ State, $M$[New York City] $\rightarrow $ City and the sentence “Yesterday in New York City”, the multi-token matcher assigns the City gazetteer type to the longest segment “New York City”. Single-token matching is searching to match any vocabulary word from a gazetteer type. In the earlier example, each word from the sentence is individually matched to the tokens in $M$, thus “New” and “York” are individually matched to both City and State, and “City” is matched only to City.

BIBREF12 research shows that both multi-token and single-token matching perform better on certain datasets. We propose to combine both methods: we tag the multi-token matches with BILU tags, and the single-token matches with a Single (S) tag. The single-token matches are used only if multi-token matches are not present. We consider that the single-token matches are high-recall low-precision, and multi-token matches are low-recall and high-precision. Thus, a combination of both works better than individually. Example sentences are: “Yesterday in New(City-B) York(City-I) City(City-L)”, and “Yesterday in York(City-S) City(City-S)” York City is marked with singles tag since $M$ does not have entities for “York City”, “York”, and “City”.

Note that gazetteer matching is unsupervised, i.e., we do not have a ground truth of correctly matched sentences for $M$. Furthermore, it is a hard task because of the many variations in writing and ambiguity of entities.

Approach ::: Gazetteer Embeddings

px

Equations DISPLAY_FORM11- shows the gazetteer embedding $\mathbf {g}_i$ computation for a token $t_i$. To compute $\mathbf {g}_i$, given a set of $m$ gazetteer types $\lbrace g^m_i\rbrace $ and spans $\lbrace s^m_i\rbrace $, we execute the following procedure:

[topsep=1pt, leftmargin=15pt, itemsep=-1pt]

Equation DISPLAY_FORM11. We embed the sets $\lbrace g^m_i\rbrace $ and $\lbrace s^m_i\rbrace $ using the embedding matrices $\mathbf {G}$ and $\mathbf {S}$. Then, we do an element-wise addition, denoted $\oplus $, of the corresponding types and spans embeddings to get a matrix $\mathbf {E}_i$.

Equation . We compute $\mathbf {A}_i$ using scaled dot-product self-attention BIBREF23, where $d$ is the dimensionality of the gazetteer embeddings. The attention contextualizes the embeddings with multiple gazetteer matches per token $t_i$.

Equation . To add model flexibility, we compute $\mathbf {H}_i$ with a position-wise feed-forward layer and GELU activation BIBREF24.

Equation . Finally, we perform max pooling across the embeddings $\mathbf {H}_i$ to obtain the final gazetteer embedding $\mathbf {g}_i$.

Approach ::: Gazetteer Dropout

To prevent the neural NER model from overfitting on the gazetteers, we use gazetteers dropout BIBREF25. We randomly set to zero gazetteer embeddings $\mathbf {g}_i$, so the gazetteer vectors that are input to the LSTM become zero. We tune the gazetteer dropout hyperparameter on the validation set.

Experiments ::: Setup

Datasets. We evaluate on the English language versions of CoNLL-03 dataset BIBREF5 and the human annotated portion of the Ontonotes 5 BIBREF6 dataset. CoNLL-03 labels cover 4 entity types: person, location, organization, and miscellaneous. The Onotonotes 5 dataset is larger and its labels cover 18 types: person, NORP, facility, organization, GPE, location, product, event, work of art, law, language, date, time, percent, money, quantity, ordinal, cardinal.

px

Gazetteers. We use the Wikidata gazetteers with types merged to the granularity of the CoNLL-03 and Ononotes 5 datasets. We filter non-relevant types (e.g., genome names, disease) and get a total of one million records. For CoNLL-03 and Ontonotes 5, the percentage of entities covered by gazetteers are 96% and 78% respectively, and percentage of gazetteers wrongly assigned to non-entity tokens are 41%

and 41.5% respectively.

Evaluation. We use the standard CoNLL evaluation script which reports entity F1 scores. The F1 scores are averages over 5 runs.

Configuration. We use the Bi-LSTM-CNN-CRF model architecture with ELMo language model embeddings from BIBREF2, which consist of 50 dim pre-trained Glove word embeddings BIBREF20, 128 dim Char-CNN BIBREF21, BIBREF1 embeddings with filter size of 3 and randomly initialized 16 dim char embeddings, 1024 pre-trained ELMo pre-trained embeddings, two layer 200 dim Bi-LSTM, and output CRF layer with BILOU (Beginning Inside Last Outside Unit) spans BIBREF22.

For the gazetteer embeddings, we use 128 dim for the embedding matrices $\mathbf {G}$ and $\mathbf {S}$, 128 dim output for $\mathbf {W}$, which yields a gazetteer embedding $\mathbf {g}_i$ with 128 dim. The parameters are randomly initialized and trained. We apply gazetteer dropout of 0.1 which we tuned on the development set; we tried values form 0.05 to 0.6.

All parameters except the ELMo embeddings are trained. We train using the Adam BIBREF26 optimizer with learning rate of 0.001 for 100 epochs. We use early stopping with patience 25 on the development set. Batch size of 64, dropout rate of 0.5 and L2 regularization of 0.1.

Experiments ::: Results

The experimental results for NER are summarized in Table TABREF20. The top part of the table shows recently published results. BIBREF14's work is using gazetteers with HSCRF and BIBREF4's work is using the Flair language model which is much larger than ELMo. BIBREF27 is the current state-of-the-art language model that uses cloze-driven pretraining. The bottom part of the table is shows our baseline models and results with included gazetteers. We experiment with the Neural CRF model with and without ELMo embeddings. Including ELMo embeddings the CoNLL-03 and Ontonotes 5, F$_1$ score improves from 92.34 to 92.86 and 89.11 to 89.32 respectively. Without ELMo embeddings the F$_1$ score improves from 90.42 to 91.12 and 86.63 to 87 respectively. We observe that GazSelfAttn relative improvements are similar with and without ELMo embeddings. We obtain slightly better CoNLL-03 F$_1$ score compared to BIBREF14 work that uses the HSCRF model, and we match the Ononotes 5 F$_1$ scores of BIBREF4 that uses a much bigger model. BIBREF14 Ononotes 5 results use subset of the dataset labels and are not comparable. Note that because of computation constrains, we did not perform extensive hyperparameter tuning except for the gazetteer dropout rate.

## Experiments ::: Ablation study

Table TABREF22 shows ablation experiments. We remove components of the gazetteer embedding model from the Neural CRF model. In each experiment, we removed only the specified component. Ablations show decreased F$_1$ score on the development and test set if any of the components is removed. The highest degradation is when single matches are removed which underscores the importance of the combining the gazetteer matching techniques for NER. We observe that match span encoding is more important for the CoNLL-02 compared to Ononotes 5 because the former has more entities with multiple tokens. Removing the self-attention shows that self-attention is effective at combining information form multiple gazetteers.

In addition, we tried moving the gazetteer embeddings to the CRF layer and using the LSTM token embeddings as attention keys but the F$_1$ degraded significantly. We experimented with adding auto-encoding loss similar to BIBREF16 and multi-head self-attention. However, we did not observe F$_1$ score improvements and sometimes small degradations.

## Conclusion

We presented GazSelfAttn, a novel approach for gazetteer embeddings that uses self-attention and match span positions. Evaluation results of GazSelfAttn show improvement compared to competitive baselines and state-of-the-art models on multiple datasets.

For future work we would like to evaluate GazSelfAttn on non-English language datasets and improve the multi-token gazetteer matching with fuzzy string matching. Also, we would like to explore transfer learning of gazetteer embeddings from high-resource to low-resource setting.