2023 IEEE 39th International Conference on Data Engineering (ICDE) | 979-8-3503-2227-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDE55515.2023.00122

Globally Aware Contextual Embeddings for Named Entity Recognition in Social Media Streams

Satadisha Saha Bhowmick
Department of Computer Science
Binghamton University

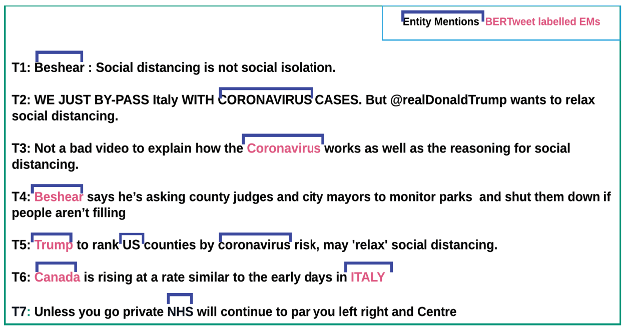
Eduard C. Dragut
Department of Computer Science
Temple University

Weiyi Meng
Department of Computer Science
Binghamton University

***Abstract*—An important task for Information Extraction from Microblogs is Named Entity Recognition (NER) that extracts mentions of real-world entities from microblog messages and meta-information like entity type for better entity characterization. A lot of microblog NER systems have rightly sought to prioritize modeling the non-literary nature of microblog text. These systems are trained on offline static datasets and extract a combination of surface-level features – orthographic, lexical, and semantic – from individual messages for noisy text modeling and entity extraction. But given the constantly evolving nature of microblog streams, detecting all entity mentions from such varying yet limited context in short messages remains a difficult problem to generalize. In this paper, we propose the NER Globalizer pipeline better suited for NER on microblog streams. It characterizes the isolated message processing by existing NER systems as modeling local contextual embeddings, where learned knowledge from the immediate context of a message is used to suggest seed entity candidates. Additionally, it recognizes that messages within a microblog stream are topically related and often repeat mentions of the same entity. This suggests building NER systems that go beyond localized processing. By leveraging occurrence mining, the proposed system therefore follows up traditional NER modeling by extracting additional mentions of seed entity candidates that were previously missed. Candidate mentions are separated into well-defined clusters which are then used to generate a pooled global embedding drawn from the collective context of the candidate within a stream. The global embeddings are utilized to separate false positives from entities whose mentions are produced in the final NER output. Our experiments show that the proposed NER system exhibits superior effectiveness on multiple NER datasets with an average Macro F1 improvement of 47.04% over the best NER baseline while adding only a small computational overhead.**

I. Introduction

In recent years, Named Entity Recognition (NER) from microblogs such as tweets has received much attention as a task of the automatic Information Extraction pipeline for a variety of analytical efforts [1]. NER deals with the extraction of contiguous strings within text that represent entities of interest in the real world. These strings (also known as surface forms as described in WNUT17 [2]), are referred to as Entity Mentions (EMs). Broadly, NER consists of two sub-tasks: 1) Entity Mention Detection (EMD) involves compiling the string variations of entities within text, and 2) Entity Typing attaches a type (e.g., person, location) to each EM for better context-specific characterization of EMs. In this paper, we focus on the challenges involving both NER sub-tasks on microblog streams, and propose a system to maximize NER effectiveness.


Entity Mentions
BERTweet labelled EMs

T1: Beshear : Social distancing is not social isolation.

T2: WE JUST BY-PASS Italy WITH CORONAVIRUS CASES. But @realDonaldTrump wants to relax social distancing.

T3: Not a bad video to explain how the Coronavirus works as well as the reasoning for social distancing.

T4: Beshear says he's asking county judges and city mayors to monitor parks and shut them down if people aren't filling

T5: Trump to rank US counties by coronavirus risk, may 'relax' social distancing.

T6: Canada is rising at a rate similar to the early days in ITALY

T7: Unless you go private NHS will continue to par you left right and Centre

Fig. 1: NER on a message stream discussing Coronavirus

Example 1. Tweets in Figure 1 have mentions (in many string variations) from six unique entities of four different entity types: '*beshear*' in T1 and T4, and '*trump*' in T5 are of type **Person**; '*italy*' in T2 and T6, '*US*' in T5, and '*canada*' in T6 are of type **Location**; '*NHS*' in T7 is of type **Organization**; and, '*coronavirus*' in T2, T3, and T5 is a disease that we categorize into the **Miscellaneous** type.

Microblog focused NER systems primarily seek to generalize the non-normative language of Microblogs. Their solutions to this problem follow an off-line paradigm, mostly employing supervised models that are specifically trained on messages sampled from Microblogs. To extract contextual information from messages, they use different surface level features – word embeddings, lexical, orthographic and semantic (POS tags, dependency trees), and sometimes even external gazetteers. The processing philosophy of these NER systems is to examine individual microblog messages once, in the order of their arrival in the stream. However these models do not make any distinction in handling the incremental and topically-correlated content generation process in Microblog streams from the offline static datasets that they use for model training.

Microblog character-limits leave inadequate context to be extracted from singular message points. This only exacerbates the challenge models face when dealing with the varied contextual possibilities, non-traditional language usage and novel tokens that populate Microblog messages. The rarity of many microblog entities in off-the-shelf lexical resources therefore still limits the performance of NER systems [3] that supplement the local context from individual messages by referencing gazetteers. The isolated message execution philosophy of existing systems has two major drawbacks that lead to sub-optimal effectiveness for NER on Microblog streams: (1) the inconsistency in detecting the same entity string across the entire breadth of a Microblog stream, i.e., the same entity mention is detected in some contexts but missed in others, and (2) the erroneous type classification of certain entities. Since type classification is heavily context-dependent, the sparse local context can often lead to incorrect entity typing. To better understand these problems, we perform NER on a message stream discussing the most prevalent conversation topic of recent times – the Coronavirus. We use a variant of the BERT language model pre-trained on a large Twitter corpus, namely BERTweet [4], that is better attuned to this noisy language setting. We fine-tune the BERTweet language model on the WNUT17 [2] training data for NER.

A Case Study. The objective of this study is to explore the performance of a deep NER system on a microblog stream and understand its limitations. We run BERTweet on a streaming dataset of 2K tweets (D_2, see Table 1 in Section VI) generated from a Coronavirus tweet stream. Although BERTweet reports state-of-the-art performance on benchmarking datasets like WNUT17, its effectiveness on this microblog stream's subset is modest, with a macro-F1 score of 43%. Not only was the extraction of novel but frequent entities inconsistent across different messages in the stream, but the F1-score also varied heavily depending on the target entity type.

Takeaways. A closer examination of the BERTweet model on the Coronavirus dataset reveals evidence for the limitations of isolated message processing. Figure 1 shows that BERTweet often missed mentions of one of the most important and frequent entities in this stream, i.e. 'Coronavirus'. Other entity mentions that came up frequently but were also missed include 'Italy' and 'US'. In addition, the F1-score of BERTweet across different entity types exhibited high variance. As a disease, entities like 'Coronavirus' or 'Covid 19' have been labelled as the Miscellaneous type, however their inconsistent detection had an impact on the final F1-score of this entity type. Apart from missed entity mentions, the frequent mistyping of entities also contribute to this. The few mentions of 'Coronavirus' recovered by BERTweet were misclassified as type 'Person'. Only 7% of the entities of type 'Miscellaneous' were ultimately recovered by BERTweet yielding a low F1-score of only 9% for this entity type. In contrast, BERTweet's F1-score for entity type 'Person' is 75%, showing how inconsistent the performance can be over different entity types when solely considering messages in isolation. BERTweet's performance also shows similar behaviour for other streaming data in our experiments. For example in dataset D_3 (see Table IV), entities of type 'Organization' like 'Justice Department' or 'Russian Government' were repeatedly mistyped.

Entities appear in many syntactic variations in the microblog ecosystem that constantly generates messages on multiple contemporaneous topics, i.e. conversation streams, evolving over time [5], [6]. The failure to consistently identify these entity mentions leads to reduced EMD performance. The high contextual variance in microblog messages with locally sparse contexts makes it difficult even for sophisticated NER models like BERT to make accurate type inferences in this setting. Also fine-tuning deep NER models for better performance with messages from newer topic streams is not always a scalable proposition, given the sheer volume and variety of conversation topics continuously emerging in platforms like Twitter. As an alternative, we propose a solution that adapts better to the shifting conversation trends of microblogs by utilizing the potential of '***Global Contextual Embeddings***' to yield better NER performance.

Collective processing with non-local context. Deep NER models operate on messages individually by first using a language model to generate token-level contextual embeddings and then a token classification head to generate outputs that encode both entity boundaries and type from the local context. Since local contexts can be inadequate for accurate inference in downstream tasks, some systems have explored collating non-local information. Akbik et al. [7] derived 'global contexts' for every unique token in its training set by performing an average pooling operation on all the local contextual embeddings generated for their mentions within a dataset. These global contexts are attached to the local embedding, when a token from the training set is encountered at test time. We argue that microblog streams are even better suited to generate global contextual representations aggregated over multiple non-local contexts, owing to their heavy entity/token recurrence. Expanding on this insight, we propose generating global contextual representations to alleviate the issues of inconsistent EMD for microblog streams in [8]. It begins with a clear delineation of local and global contexts in EMD computation for a streaming setting. Local contexts suggest entity candidates. It then aggregates local representations of candidate mentions to generate global contextual embeddings and distinguish entities from false positives. In essence, [8] shifts the focus of EMD computation beyond the confines of a single sentence to *collectively process* mentions of an entity across the entire span of a stream.

In [8], we first presented the potential of collective processing on microblog streams for the EMD subtask. In this paper, we lay out a system that (i) addresses the unique challenges when bringing Entity Typing into the fold, and (ii) sets a collective processing pipeline for the complete NER task. The locally limited context in microblog messages results in inconsistent EMD and mistyping of entity mentions. Aggregating local contextual embeddings across all syntactic variations of an entity candidate has yielded improvements for EMD, however simply extending the framework to Entity Typing leads to misleadingly aggregating contexts from mentions that have the same surface form, but correspond to different entities. For example, the candidate string '*Washington*' can correspond to entities of two different types – Person (the American president) or Location (the American state) – depending on its context. Since Entity Typing is inherently context dependent, arbitrary aggregation of local contexts without factoring in such surface form ambiguity reduces the benefits of collective processing and generates incorrect entities. The solution we propose here is specifically mindful of this issue.

Approach Overview. In this paper we propose the NER Globalizer system that demonstrates how to effectively mine global contextual embeddings for NER on microblog streams. We begin with a traditional NER step of isolated message processing using a fine-tuned BERTweet model, but only to generate a set of seed surface forms in which entity candidates appear within a conversation stream. The language model also produces token-level contextual embeddings. This constitutes our '**Local NER**' phase. Although Local NER encodes both boundary information and associated entity type in its token-level labels, these outputs can be unreliable owing to the scarce local context from which they are generated. We first follow up Local NER with model agnostic occurrence mining that yields additional mentions of these surface forms that were previously missed. This is crucial for correcting inconsistent EMD. Next, an '*Entity Phrase Embedder*' uses the token-level contextual embeddings from Local NER to generate a unified embedding for individual mention phrases of seed candidate strings. The phrase embedder is trained using contrastive learning such that surface form mentions of the same type congregate closely in the candidate embedding space but are distant from mentions of other entity types. Up until this point the embeddings we generate are still locally confined. Based on the local mention embeddings, we draw an optimal clustering to group mentions of a surface form into different (potential) entity types. Then we use a learned pooling function to generate a global candidate embedding for each candidate cluster. Since each cluster is arguably an entity type, this embedding is a global representation of the corresponding candidate-type pair. Finally an *Entity Classifier* is trained to use a cluster's global embedding and label it as one of the preset entity types. The classifier also helps us separate false positives from valid entities. The steps following Local NER up to the labeling of candidate clusters constitute what we call '**Global NER**'.

NER Globalizer reduces the inconsistencies in traditional NER processing significantly. We test the system on several existing benchmarks and outperform a locally limited BERT-fine tuned NER model, which itself is a strong NER baseline, by 47.04% on Macro F1 score. On average, the inclusion of Global NER reports an average F1 improvement of 11.49% for type Person and 22.58% for type Location. For entity types that are prone to greater mislabeling the average F1 improvement is even more remarkable, about 174% for Organization and Miscellaneous. This work has also lifted the performance ceiling on popular NER datasets WNUT17 [2] and BTC [9]. Our paper makes the following contributions:

- We propose a novel NER system that provides a clear delineation between the scope of local and global NER computation for microblog streams. With Global NER we demonstrate how to effectively aggregate contextual information for better NER performance in this setting. It supports continuous and incremental execution which is in tune with the message generation process of streams.
- Local NER is decoupled from Global NER steps. This allow us to test the hypothesis that aggregating contextual embeddings leads to better NER performance in the streaming setting and it makes the the system more nimble to incorporate future local NER designs.
- In our experiments, we evaluate the proposed framework on both in-house streaming Twitter datasets and third party datasets. We not only report superior performance for entity typing but also bring in improvements for EMD.

The code for NER Globalizer and experimental datasets are available at https://github.com/satadisha/collective_NER.

II. Related Work

The principal issues impacting an NER performance in the microblog streaming setting are: a) its ability to consistently detect entity mentions across the diverse contexts in which they appear, and, b) resolving entity mentions to correct entity types. [2] posits that the lack of annotated data from this domain, and the difficulty in identifying emergent entity forms are the major factors contributing to these issues. The NER literature features a wide range of supervised systems that demand a training set with token level annotations that encode both boundary and associated type information. Usually a variant of the BIO-*type* (Beginning-Inside-Outside) encoding [10] is used to prepare these annotations. Broadly, these systems either operate on handcrafted linguistic features in a non-neural system or use deep neural networks with minimal feature engineering. The first category of systems, like [11], [12], [13] recreates an information extraction pipeline starting with POS-taggers to extract and feed semantic features to a CRF based entity boundary segmentation module. In some systems, the feature set is enhanced with word embeddings or gazetteers to supplement the limited contextual possibilities of microblogs and the rare tokens that inhabit the medium [14], [15], [16], [17], [18], [19], [20].

Many **Deep NER systems** [21], [22], [23], [24], [25], [26], [27], [28], [29], [30] have recently been adopted for the sequence labeling task of NER. The recent WNUT shared tasks [31], [2] delve into a variety of Deep NER systems for Tweets. Aguilar et al. [3]– a BiLSTM-CNN-CRF architecture– performed best at the WNUT17 task. More recently BERTweet [4] – a BERT language model pre-trained on a large Twitter corpus– reported the most effective language model when performing a variety of language tasks. For our Local NER step, we adopt the pre-trained BERTweet model and fine-tune it with the WNUT17 training set. [32], [33] examine the cross-domain transferability of DNN features learned from the more abundantly labelled well-formatted corpora to overcome the lack of annotated data from the microblog domain.

Other alternatives include TwiNER [34], an unsupervised approach using collocation features from the Microsoft Web N-Gram corpus [35] or joint modeling of NER [36], [37], [38], [39] with other information extraction subtasks.

Using global context. The idea of aggregating non-local contextual information for NER has only been considered in [7] for the microblog setting. However a similar rationale is encountered in some of the recent Deep-NER pipelines for

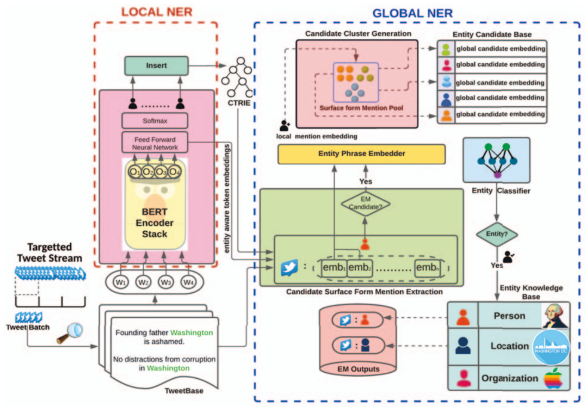


Fig. 2: NER Globalizer System Architecture

documents. Unlike a stream of tweets produced by multiple authors, documents are structurally more cohesive with well-formatted language. But both repeat entities and tokens through the collective span of their contents. Document-level NER systems like HIRE-NER [40] utilize this tendency to distill non-local information for each unique token, from the entire scope of the document, using a memory-structure and append them to sentence-level contextual embeddings before an NER decoder draws final output labels. DocL-NER [41] additionally includes a label refinement network to enforce label consistency across documents and improve NER results. Therefore in our experiments, we consider [7], [40] and [41] as baselines for Global NER and compare their performance alongside NER Globalizer to evaluate how effectively global information is compiled in each approach.

The idea of collectively viewing mentions to distinguish entities from false positives has been explored for the EMD subtask in TwiCS [42] and EMD Globalizer [8]. TwiCS [42] uses a shallow syntactic heuristic to identify entity candidates and then generates syntactic support from their mentions to distinguish legitimate entities from false positives. EMD Globalizer moves beyond narrow syntactic assumptions and makes robust constructions towards the computation of global contextual embeddings in the streaming space. But neither systems provide an implementation for the complete NER pipeline by leaving out the crucial task of type classification.

Though NER Globalizer relies on a robust language model to fully leverage the local context of individual messages, it ultimately differs from the traditional NER setup by: i) aggregating local contexts for entities to incrementally build representations that are resilient to local noise and yield better contextualization from the entire scope of the stream, and, ii) build solutions that can exploit both the local connectivity of named entities to other tokens within the immediate context and the ('collective') global contexts of entities within the stream, in order to better resolve entity definitions. Our approach in this work not only maintains a continuous and incremental computation setup, which is conducive to processing streams, but also improves upon the collective processing setup of EMD Globalizer by laying out all the necessary considerations, like handling surface form ambiguity, for appropriately aggregating non-local contexts to solve both EMD and Type Classification in an integrated pipeline.

III. System Overview

In Figure 2 we illustrate the overall architecture of NER Globalizer that facilitates a continuous execution setup, sustaining over multiple iterations, as long as new messages are produced in a tweet stream. Each iteration consists of a batch of incoming tweets thereby discretizing the evolution of the underlying stream. Also for entity typing we limit the classification to one of L preset entity types. In the current implementation we test with four common entity types: Person (*PER*), Location (*LOC*), Organization (*ORG*) and Miscellaneous (*MISC*).

We commence with a few core definitions for clarity:

Definition III.1. *Entity Candidate* corresponds to canonical strings within a message stream that likely represent a real-world entity of a specific type. For example, in Figure 1 '*beshear*' is an entity candidate of type Person that corresponds to Governor of Kentucky Andy Beshear.

Definition III.2. *Candidate Surface Forms* correspond to canonical strings in which candidates appear within a stream. For example, in Figure 1 we consider '*beshear*' to be the surface form for the corresponding entity candidate. Candidate Surface Forms may be ambiguous, such as '*washington*' that may refer to different entity candidates (of type *person* or *location*), depending on the context in which they appear.

Definition III.3. *Candidate Mention* is an individual reference to a candidate in a message. Candidate mentions can appear as many variations of the canonical surface form for a candidate within a stream. In Figure 1 we find three mentions of the entity candidate '*coronavirus*' in tweets T2, T3 and T5.

An execution cycle in NER Globalizer proceeds as follows:
(1) The Streaming module fetches a stream of tweets, on a particular topic, using the Twitter streaming API.
(2) First we run a batch of tweets through a pre-trained BERT encoder fine-tuned on the downstream NER task, one sentence at a time. This traditional execution setup is the **Local NER** step. The extracted phrases are registered as seed candidate surface forms in which entities are likely to populate the stream. Also the token-level outputs of the language model's final layer, before token classification, are stored for every tweet in the batch, as '*entity-aware token embeddings*'.
(3) Next we initiate **Global NER** that is decoupled from the Local NER step and would therefore not require any alteration upon a different choice of language model for Local NER. It comprises of a few additional steps:

(i) First, a scan of the tweet batch extracts all mentions of the seed candidate surface forms discovered so far. This involves finding mentions that were missed by local NER in the previous step, along with the ones that were already found.

(ii) For every mention we generate a contextual embedding based on the immediate local context. Here the token-level contextual embeddings generated during Local NER are passed through a Phrase Embedder to construct an unified embedding for the entire mention phrase.

(iii) For type classification, we consider that the same surface form may correspond to entities of different types depending on the context in which it is mentioned. To resolve this ambiguity we use the local embeddings of surface form mentions to derive an optimal clustering of type manifolds in the mention representation space. Each cluster of mentions corresponds to an entity candidate likely belonging to one of the pre-set entity types.

(iv) Local contextual embeddings of every mention in a candidate cluster are then aggregated using a learned pooling function to generate the candidate's global embedding. The mention subspace for a candidate surface form and the resulting global embeddings can be incrementally updated by adding local embeddings into the pool as new mentions of the surface form appear.

(v) Finally global candidate embeddings are passed through the entity classifier to separate entities from false positives (non-entities). This is a multi-class classification module with outputs belonging to one of $L + 1$ classes – we consider an additional non-entity class along with the preset entity types. Mentions of candidates labelled as one of the L entity types are produced in the system's final NER outputs for the tweet batch along with the type labelled during classification.

We elaborate on these steps in later sections.

IV. Local NER

The first step in NER Globalizer is Local NER. This is a traditional NER setup where every tweet-sentence in a batch of tweets is processed individually to detect entities from local context. The NER process here is a sequence labeling task generating BIO token labels which encodes the position of a token relative to its nearest entity boundary as well as the associated entity type. The state-of-the-art NER technique uses a language model (Transformer encoder [43], [44] or BiLSTM [29], [30]) to study the immediate context and generate token-level contextual embeddings. In practice the language model is pre-trained unsupervised learning of language representations from large text corpora. The token-level contextual embeddings are obtained at the penultimate layer of the deep neural network. For NER, pre-trained language models are appended with a multi-class classification head to generate the token-level output labels. The entire model is then fine-tuned end-to-end using an annotated training set to optimize performance for the downstream NER task. Therefore we interpret the contextual embeddings, post fine-tuning, as local entity-aware embeddings, extracted from the context of a single input.

Objectives: For a targeted stream of tweets, Local NER aims to: (1) generate a set of candidate surface forms from mentions in individual sentences, and (2) encode local entity-aware information for every token in a sentence.

The Local NER module acts as a weak labeller in our system. Despite the noisy outputs, it is essential for generating a good set of seed candidate surface forms that will be directly utilized in later steps and are therefore crucial for maintaining good recall. In NER Globalizer we use a state-of-the-art language model – BERTweet [4] – to construct an effective Local NER module. BERTweet is the first language model trained on a large-scale corpus of English Tweets. This system has the same architecture as BERT$_{\text{base}}$ [44] but uses the RoBERTa [45] pre-training procedure for better performance.

To fine-tune the language model for NER, a feed forward neural network layer and a softmax layer are added on top of the last Transformer encoder. The fine-tuning is conducted using the WNUT17 training set and is repeated five times with randomly initialized seeds. The reported performance is the average of five test runs. We use the pre-trained BERTweet model publicly available at the Hugging Face model hub that amasses a large collection of pre-trained language models catering to a variety of downstream NLP tasks. Although the WNUT17 dataset consists of six entity types, the type coverage for NER-Globalizer is limited to the four types mentioned in Section III. Hence we group instances of types 'Product', 'Creative-work' and 'Group' as the 'Miscellaneous' type.

The set of seed surface forms derived during the entity labeling of Local NER are stored in a **CandidatePrefixTrie** (**CTrie** for short). CTrie is a prefix Trie forest used for efficient indexing. During Global NER, CTrie facilitates easy lookups to find all mentions of these discovered candidate strings. Another data structure produced at the end of Local NER is **TweetBase** that maintains individual records for every tweet sentence indexed by a (tweet ID, sentence ID) pair and a list of detected mentions that get updated after Global NER.

In a traditional execution set-up this would conclude the NER process. However, as explained in Section I, for Microblogs the NER outputs at this stage can be erroneous. The length constraint imposed on Microblog content leaves inadequate context to be found in individual messages. This only exacerbates the challenge that models face when dealing with the varied contextual possibilities, non-traditional language uasge and novel tokens populating the conversations in Microblogs. This results in the same entity being inconsistently extracted or mistyped across the breadth of a Microblog stream. To remedy these issues we introduce the Global NER module.

V. Global NER

Local NER produces a set of candidate surface forms that are likely to represent entities within a message stream as well as local contextual embeddings of sentence tokens. But these representations and inferences drawn from them are locally limited and can be error-prone. To address these limitations, the Global NER module shifts the focus of entity extraction from being limited to the confines of a single sentence to viewing mentions of the entity candidates collectively in a type manifold or embedding subspace across the span of the entire stream. This idea of representing an entity candidate collectively through its mentions is called '***collective processing***'. We make robust constructions in this space supporting incremental computations that evolve with the stream itself.

Objectives: The Global NER step addresses issues pertaining to both entity mention detection and typing:

1. Removal of False Negatives: False Negatives arise when Local NER fails to tag entity mentions in some sentences. In Figure 1, *coronavirus* in T2 and T5 is a false negative.

2. Removal of False Positives: False Positives happen when Local NER extracts non-entity phrases as entities.

3. Correction of Partial Extraction: Partial extractions happen when a part of an multi-token entity string is left outside of a labeling window in BIO format. Correcting such partial extractions improves both recall and precision.

4. Correction of Mistyping: Classifying an entity mention to incorrect entity type is registered as a false negative for the target type and also impacts NER performance.

Global NER consists of four steps that we will now describe.

A. Mention Extraction

The objective of the mention extraction step is to discover all mentions of candidate surface forms registered in the CTrie during Local NER. This step is crucial for finding missed mentions of candidates that would later on be verified as entities. Hence it has a direct impact on recall. But additional mentions are also useful in deriving better collective context for candidates which even though have been tagged in Local NER would ultimately be false positives. Hence we also posit this step to be important for precision improvement.

Empowered by the candidate surface forms seeded during Local NER in the CTrie, we reduce the mention extraction process to that of a simplified lookup in the CTrie. The module analyzes every token in a tweet sentence, in conjunction with a CTrie traversal. With a case-insensitive comparison of tokens with CTrie nodes, this results in two possibilities:

(i) A token that matches a node on the current CTrie path, when cases are ignored.

(ii) A token matching no node in current path.

We check if a token forms a mention of a candidate surface form alone or together with up to k following tokens.

The extraction process scans a tweet-sentence and identifies the set of longest subsequences that match with candidate surface forms in the CTrie, mentions extracted during Local NER are verified, and sometimes corrected. For example, if Local NER finds only a partial excerpt '*Andy*' of the candidate '*Andy Beshear*' in a tweet, but nonetheless recognized the entire string in other tweets, the candidate surface form ('*andy beshear*') will be registered in the CTrie. This partial extraction can now be rectified to the complete mention. The process is syntax agnostic. It initiates a window that incrementally scans through a sequence of tokens. In each step it checks:

a) whether the subsequence within the current scan window corresponds to an existing path in the CTrie. If true, it implies that the search can continue along the same path, by including the token to the right within the window in the next iteration.

b) whether the node on this path, matching the last token of the subsequence, refers to a valid candidate. If true, we record the subsequence as the current longest match.

In case of a mismatch, the last matched subsequence within the current window is stored, and the process then skips ahead initializing a new window from the position of the next token. The search for a new match starts along a new CTrie path. However, if the last search had failed to match with any of the existing CTrie paths, the new window starts from the token that is to the immediate right of the first token in the previous window. The process is repeated until all tokens are consumed. In the end we obtain a set of mention variations for each candidate surface form in the CTrie.

B. Local Mention Embedding Generation

Collective processing views a candidate as an aggregate of its mentions within a stream. To achieve this, we first need to derive semantically meaningful representations of individual mentions, in their local context, and then aggregate these representations to derive a candidate's global embedding. Given that mention phrases may have variable number of tokens, we need to combine the token-level embeddings coming out of Local NER into a unified, fixed-size embedding of the entire phrase. This is the role of the ***Phrase Embedder***.

To generate mention phrase embeddings, we refer to two lines of work. First is the literature on generating multi-token embeddings at a sentence level for *Semantic Textual Similarity* (STS) tasks [46], [47] and second is the idea of *Supervised Contrastive Learning* [48].

Design of Phrase Embedder. We adopt a modification of the '*siamese network structure*' from SBERT [49]. We use Local NER to generate token-level embeddings as the principal component of the mirrored subnetworks in a triplet network structure. Then we use average pooling to combine token-level representations into an average embedding. Next we apply $l2$-normalization on the average embeddings and pass the unit-norm vectors through a final dense layer. Our experiments reveal that adding the normalization step leads to better performance. The local mention embeddings ($local\_emb \in \mathbb{R}^d$) from token-level embeddings can be computed using one of the Entity Phrase Embedder sub-networks as

$$pooled\_emb = \frac{1}{|T|}\sum_{j=1}^{|T|} token\_emb_{T_j} \tag{1}$$

$$\widehat{pooled\_emb} = \frac{pooled\_emb}{|pooled\_emb|} \tag{2}$$

$$local\_emb = W_{ff}\widehat{pooled\_emb} + b_{ff} \tag{3}$$

where T denotes the set of tokens in the candidate phrase, $token\_emb_{T_j} \in \mathbb{R}^d$ is the contextual embedding of the j-th token in T, generated by the Local NER engine. The weight matrix $W_{ff} \in \mathbb{R}^{d\times d}$ and bias $b_{ff} \in \mathbb{R}^d$ are trainable parameters from the mirrored sub-networks. However, unlike SBERT, the gradient computation is not backpropagated all the way to the BERT engine of Local NER in our case. Instead, the weights fine-tuned during Local NER remain frozen in our siamese network and only the weights of the layers following it are updated. This is because the Local NER module's role in our framework is to produce (Local) EMD results for which it had already been optimized. The rest of the sub-network however work on the 'entity-aware' token embeddings to produce an optimal phrase embedding.

Training the Phrase Embedder. Supervised contrastive estimation [48], [50], a well-explored domain of representation learning, learns embeddings by optimizing instance-level discrimination. Contrastive estimation is typically trained on datasets where individual records consist of an anchor example for which learned embeddings are tuned to be closer to a set of positive examples in the representation space and distant from a set of negative examples. We deem this to be suitable when choosing our objective function as the purpose of Phrase Embedder is not only to capture the local contextual relations between mention tokens but also to produce a mention embedding that aids in the subsequent clustering step. The clustering is essential to map mentions correctly to the candidate they represent and resolve the issue of candidate surface form ambiguity. We elaborate on this issue in Section V-C. As such the phrase embedding should be such that, when mapped to the representation space populated by all mentions having the same surface form, mentions of the same type should congregate together in a manifold that is well-separated from mentions of other types. This would not only lead to well separated candidate clusters but also generate accurate global embeddings for type classification. We train the Embedder with Triplet loss [51] and Soft Nearest Neighbour loss [52], which are both special cases of contrastive loss.

Triplet Objective Function. Triplet loss uses one positive (p) and negative (n) example per anchor. p is another instance from the same class as the anchor (a) while n is from a different class. Mathematically, we minimize the following loss such that the distance between the anchor and positive example is less than its distance with the negative example.

$$max(\|local\_emb_a - local\_emb_p\| - \|local\_emb_a - local\_emb_n\| + \epsilon, 0) \tag{4}$$

$local\_emb_x$ is the embedding generated for $a/p/n$, $\|.\|$ is a distance function, and ϵ is the margin value. We choose the $Cosine$ distance function and set the margin value to be 1 to encourage orthogonality between the mentions of a surface form that ultimately correspond to different entity types.

Soft-Nearest Neighbour Objective Function. For using multiple positive and negative examples per anchor, noise-contrastive optimization is extended to the Soft Nearest Neighbour Loss [53], [52]. The loss measures entanglements of type manifolds which characterizes how close pairs of mention representations of the same entity (type) are, relative to representations of surface form mentions of different entities.

$$-\frac{1}{b}\sum_{i\epsilon 1\ldots b} log(\frac{\sum_{j\epsilon 1\ldots b; j\neq i; y_i=y_j} e^{\frac{-\|local\_emb_i - local\_emb_j\|}{\tau}}}{\sum_{k\epsilon 1\ldots b; k\neq i} e^{\frac{-\|local\_emb_i - local\_emb_k\|}{\tau}}}) \tag{5}$$

The loss computation is approximated over mini-batches. Intuitively this loss is the negative log probability of sampling a mention j from the same candidate cluster (type manifold) as the anchor mention i. The hyperparameter *temperature* (τ) controls the relative importance given to the distance between pairs of mentions. [52] suggests setting the temperature to a lower value for neighbouring points from the same class to have more importance in tuning the representation space than distant point from other classes. Here we use $Cosine$ distance and typically start with smaller values of τ that we fine-tune.

Our experiments reveal that training the Phrase Embedder with Triplet loss generates more accurate candidate clusters and better performance for the Entity Classifier. Hence we use this loss for the production version of NER Globalizer. We provide more details on the training process of Phrase Embedder with contrastive estimation in Section VI.

C. Candidate Cluster Generation

At this point NER Globalizer has extracted all mentions of the various candidate surface forms in the CTrie along with their local contextual representations in the embedding space. But we need to consider the issue of '*ambiguous candidate surface forms*'. Depending on the context in which they appear, mentions of the same candidate surface form can in fact refer to different entities or non-entities. Consider in Figure 1 the commonly used pronoun us that has the same surface form as the commonly used abbreviation of United States i.e. '*US*'. When aggregating local mention-level representations to generate global candidate embeddings, it is crucial to first map mentions to their appropriate candidate (type) manifold. Hence we use contrastive estimation when generating local mention representations to leverage better separation among mentions of different candidate types that share the same surface form. We follow this up by deriving an optimal clustering over the mention embeddings of every unique candidate surface form discovered during Local NER. This reveals the different underlying candidates that share the representation space through the same candidate surface form. For example, in case of the surface form 'us' we would ideally have two separate clusters one corresponding to the non-entity pronoun and the other corresponding to the country.

The candidate cluster generation step serves two purposes in the execution cycle of NER Globalizer:

1. By studying the relative distance of a mention representation to that of other mentions in the embedding space, this step associates mentions with proper candidate-type thereby rectifying the issues of mistyping with Local NER.

2. For mentions missed by Local NER but discovered during Mention Extraction there are no types attached since Local NER produced the 'O' label for their tokens. The clustering step also makes the optimal association of these mentions with candidates to properly leverage their discovery for recall gain.

The number of candidate clusters per surface form is unknown in advance. Hence we use agglomerative clustering [54] which does not need to know the number of clusters. We use $Cosine$ distance with average linkage for the clustering process to leverage the separation that was optimized when the training of local mention embeddings. Agglomerative clustering needs a distance metric threshold beyond which clusters are considered separate during the bottom-up merging process.

For $Cosine$ distance a value of 1 indicates orthogonality that we set as the margin value of our Triplet loss. Hence we tune the appropriate threshold to be less than 1. We also consider including a clustering loss when learning local mention representations by looking into some recent literature on Deep Clustering [55], [56], [57]. However these methods also require that the number of clusters be fixed and known a priori. Since the candidate clusters corresponding to a candidate surface form violate this, we keep the objective function for representation learning limited to instance-level contrastive learning and the subsequent clustering step unsupervised. Both the representation space for a candidate surface form and the clusters drawn from its mentions are updated as and when new mentions arrive in the stream.

D. Entity Classifier

The local embeddings of individual mentions in a candidate cluster are limited to the context of the sentence containing it. We add these local embeddings to the candidate's record in a data structure called the CandidateBase, which maintains an entry for every candidate discovered for a stream during the previous step. In essence, every candidate cluster corresponds to a unique entity candidate in the CandidateBase. Next, a pooling operation on all the local mention embeddings within a candidate cluster gives the '***global candidate embedding***'.

$$a_j = W_a^T local\_emb_j + b_a \tag{6}$$

$$w_j = \frac{exp(\alpha_j)}{\sum_{k=1}^{\|\eta\|} exp(\alpha_k)} \tag{7}$$

$$global\_emb = \sum_{j=1}^{|\eta|} w_j local\_emb_{\eta_j} \tag{8}$$

where the weights $W_a \epsilon \mathbb{R}^d$ and bias term b_a are learnable parameters, η denotes the set of mention-level local embeddings for a candidate, and w_j is the weight of the local embedding ($local\_emb_{\eta_j} \epsilon \mathbb{R}^d$) of the j-th mention in η. The embedding $global\_emb$ is global in the sense that it generates a consensus representation from all contextual possibilities in which a candidate appears in the stream.

The global candidate embeddings are fed to a network of multiple dense layers with ReLU activation and a softmax output layer. This module is the ***Entity Classifier***. It is trained to determine if a candidate belongs to one of $L + 1$ classes – an entity of one of the L pre-set types or a non-entity.

The classifier is trained using global embedding records of labelled entities from D_5 (see Table I). We find that adding the non-entity class is effective in separating the L entity types from non-entities. However, D_5 is annotated only with entities. Hence to include non-entities and their global embeddings in the training set of the classifier, we run BERTweet instantiated EMD Globalizer on D_5 and curate a set of seed non-entities.

VI. Experiments

We conduct extensive experiments to test NER Globalizer for named entity recognition in tweets. First we test the effectiveness of our Global NER technique by calculating the improvement it brings on the standalone Local NER module in our implementation, i.e. BERTweet [4]. In addition, we also use three existing Global NER systems as baselines against NER Globalizer to compare how effectively global context is mined and used for NER in each system. We implemented NER Globalizer in Python 3.8 and executed it on a NVIDIA Tesla T4 GPU on Google Colaboratory.

TABLE I: Twitter Datasets

Dataset	Size	#Topics	#Hashtags	#Entities
D_1	1K	1	1	283
D_2	2K	1	1	461
D_3	3K	3	6	906
D_4	6K	5	5	674
D_5	3430	1	1	-
WNUT17	1287	-	-	-
BTC	9553	-	-	

Datasets: The datasets used to evaluate NER Globalizer are listed in Table I. D_1-D_4 are **streaming datasets** that contain messages crawled directly from Twitter streams. The topics covered here are Politics, Sports, Entertainment, Science and Health, with (D_2) curated from a Covid-19 tweet stream. Having datasets that are subsets of tweet streams helps preserve their natural topic-specificity that often repeats a finite set of entities. This is used in Global NER to collectively process candidates, without making the analysis biased towards a particular topic. In real-world deployment, a topic classifier [58] could precede an NER tool launched for streams.

Other than the four streaming datasets D_1-D_4, two datasets popular for NER benchmarking, WNUT17 [2] and BTC [9], are also used in our evaluation. These are **non-streaming datasets** curated with a random sampling of tweets. Although they do not characterize the application setting for NER Globalizer, we use them to gauge the system's effectiveness against established benchmarking standards.

We use dataset D_5, a collection of 3.4K tweets from a single tweet stream to generate entity candidates and train the supervised components of our Global NER setup, namely the Phrase Embedder and Entity Classifier. The datasets are labelled with entity mentions along with their associated entity types. Since our system is also designed to recognize and separate non-entities, we need representative non-entities to complete the training of NER Globalizer. To this end, we run BERTweet instantiated EMD Globalizer on D_5 and generate a set of seed non-entities that we use to both learn mention embeddings for non-entities and train the Entity Classifier.

Performance Metrics: We use Precision (P), Recall (R) and F1-score for each of the four pre-set entity types to evaluate NER effectiveness. Further, we also use Macro-F1 (**F1 (Entity)** in the WNUT17 shared task [2]) to make summary comparisons of NER performance across different systems in our evaluation. Note that a correct NER detection requires both EMD and Entity Typing to be handled correctly.

Local NER Baselines: We use two state-of-the-art NER systems that exhibit good performance on the WNUT17 dataset as our Local NER baselines. Aguilar et al. [3] was the best performing system among those featured in the WNUT17

NER challenge. It uses a BiLSTM-CNN-CRF pipeline within a multi-task learning pipeline that learns higher-order character-level, token-level and lexical feature representations. BERT-NER [44] is the seminal BERT language model by Devlin et al. that demonstrated the efficacy of contextual embeddings for a variety of language tasks. Here we fine-tune it for NER.

Global NER Baselines: We use three baselines: Akbik et al. [7] and two Document NER systems, HIRE-NER [40] and DocL-NER [41], to test Global NER. We compare their performance with NER Globalizer on our Twitter datasets. Despite the structural differences between documents and tweet streams, both exhibit topically correlated cross-sentence information. Much like the intuition of aggregating local embeddings for global representations in NER Globalizer, HIRE-NER and DocL-NER use a similar observation in their design for collecting global information. Hence their Global NER techniques are appropriately considered in our evaluation. When executed on our datasets, both systems treat messages in a stream as composite content, much like a document.

Training the Phrase Embedder: The Phrase Embedder during Global NER combines token-level contextual embeddings of a candidate surface form's mention, obtained during Local NER, into a unified local embedding for the entire mention phrase. The Phrase Embedder is trained through contrastive estimation with two different objective functions that require their own dataset curation and data augmentation process.

1. ***Mention Triplet Mining***: Training with triplet loss requires a dataset where individual records are triplets of an anchor, a positive and a negative example. To this end, we use the entities and non-entities in D_5 as our candidate set and gather their corresponding mention sets. For every mention in the mention set of an annotated candidate, a positive example would be another mention from the same set while a negative example would be a mention of a candidate that shares the same surface form but is of a different type. We want to increase the separation between the anchor and the negative example such that the mention-level phrase embeddings produced in this step aids the clustering step over mentions that have the same surface form but need to be separated based on the entity type to which they correspond. Note that not every surface form resides over all $L+1$ types that we consider during clustering and classification. In such a case, we augment the dataset by collecting negative examples from mention sets of candidates that do not share the same surface form and belong to a different type. Here the objective is to train the Phrase Embedder to yield better inter-type separation among candidate mentions, when generating local mention-level embeddings. Following this procedure, we obtain 15.77 million triplets from D_5 to train the Phrase Embedder.

2. ***Mention Cluster Mining***: Using Soft-NN Loss requires a dataset where individual records are mentions of entities or non-entities, whose distances from mentions within the same candidate manifold, i.e. positive examples, are reduced, in contrast to other mentions of its surface form that correspond to different candidates. Even in this case we use data augmentation when considering mentions of surface forms that do not reside over all $L+1$ types. In these cases we augment mentions of one randomly chosen candidate from each remaining type. To this end, we create a dataset of 9134 records, each consisting of a mention of an entity or a non-entity in D_5 along with its positive and negative example set.

During training, we use Adam optimizer [59] with a fixed learning rate of 0.001 and maintain a 80-20 train-to-validation split. Given the vastly different dataset sizes for the two types of training, we set a batch size of 2048 for training with the Triplet objective function, and a batch size of 64 for the Soft NN objective function, in order to optimize the overall training time. For regularization we use both early stopping and weight decay and also add batch normalization. We train over 200 epochs and compute performance on the validation set after each training epoch to save the best model checkpoint but we also enforce early stopping after 8 continuous epochs.

TABLE II: Training of Phrase Embedder and Entity Classifier

Objective Function	Dataset size	Training Loss	Validation Loss	Entity Classifier Validation Macro F1
Triplet	15.77 M triplets	0.0012	0.0015	92.8%
Soft NN	9134 candidate mentions	0.3718	0.376	77.3%

TABLE III: NER Globalizer vs. Local NER systems

Dataset	NER System	Entity Type (F1 Score)				Macro F1
		PER	LOC	ORG	MISC	
D_1	NER Globalizer	0.84	0.87	0.5	0.39	0.65
	Aguilar et al.	0.22	0.24	0.22	0.07	0.19
	BERT-NER	0.65	0.52	0.2	0.13	0.38
D_2	NER Globalizer	0.82	0.68	0.54	0.58	0.66
	Aguilar et al.	0.57	0.59	0.19	0.05	0.35
	BERT-NER	0.71	0.54	0.2	0.08	0.38
D_3	NER Globalizer	0.9	0.81	0.71	0.51	0.73
	Aguilar et al.	0.44	0.56	0.46	0.14	0.4
	BERT-NER	0.65	0.65	0.12	0.15	0.39
D_4	NER Globalizer	0.83	0.91	0.68	0.71	0.78
	Aguilar et al.	0.53	0.68	0.27	0.07	0.39
	BERT-NER	0.65	0.68	0.38	0.41	0.53
WNUT 17	NER Globalizer	0.76	0.71	0.52	0.44	0.61
	Aguilar et al.	0.37	0.28	0.2	0.14	0.25
	BERT-NER	0.52	0.42	0.22	0.35	0.38
BTC	NER Globalizer	0.7	0.71	0.48	0.43	0.58
	Aguilar et al.	0.35	0.41	0.15	0.06	0.24
	BERT-NER	0.63	0.59	0.15	0.22	0.4

Training Entity Classifier: Upon training the Phrase Embedder, we use it to generate local embeddings of mentions in the ground truth clusters of entities/non-entities labelled in D_5. Next, for each ground truth cluster, the local embeddings are aggregated using weighted pooling to generate a global candidate embedding that is then passed through the rest of the Entity Classifier network to generate the final class (type) label. The learned pooling operation and the classification network are trained end-to-end to optimize the final NER Classification performance. We obtain 1391 candidates from D_5 and use a 80-20 training-to-validation split to train the Entity Classifier over 200 epochs. We use Adam optimizer

TABLE IV: **Ablation Study**: Effectiveness and Execution Time (in seconds) with NER Globalizer

Dataset	Entity Type	Local NER				Global NER				F1 Gain	Time Overhead
		P	R	F1	Execution Time	P	R	F1	Execution Time		
D_1	ORG	0.85	0.13	0.22	33.16	0.55	0.45	0.50	34.32	121.72%	1.16
	MISC	0.19	0.09	0.12		0.56	0.30	0.39		219.87%	
	LOC	0.69	0.78	0.73		0.94	0.83	0.87		20.39%	
	PER	0.82	0.68	0.74		0.94	0.76	0.84		13.58%	
D_2	ORG	0.5	0.15	0.23	40.23	0.55	0.53	0.54	42.58	133.92%	2.35
	MISC	0.16	0.07	0.09		0.60	0.56	0.58		494.83%	
	LOC	0.49	0.73	0.59		0.76	0.62	0.68		16.46%	
	PER	0.70	0.82	0.76		0.77	0.87	0.82		8.17%	
D_3	ORG	0.68	0.1	0.18	58.6	0.69	0.88	0.77	62.18	343.63%	3.58
	MISC	0.30	0.22	0.25		0.68	0.54	0.6		137.14%	
	LOC	0.66	0.68	0.67		0.82	0.89	0.84		27.43%	
	PER	0.81	0.85	0.83		0.90	0.92	0.91		9.69%	
D_4	ORG	0.89	0.23	0.37	230.75	0.66	0.7	0.68	237.53	85.87%	6.78
	MISC	0.57	0.34	0.43		0.89	0.59	0.71		66.6%	
	LOC	0.76	0.81	0.78		0.89	0.93	0.91		15.99%	
	PER	0.69	0.76	0.72		0.83	0.84	0.83		15.44%	
WNUT17	ORG	0.26	0.16	0.20	24.40	0.6	0.45	0.52	26.15	160%	1.75
	MISC	0.49	0.24	0.33		0.52	0.39	0.44		38.34%	
	LOC	0.47	0.52	0.49		0.82	0.63	0.71		44.31%	
	PER	0.76	0.62	0.68		0.83	0.69	0.76		11.76%	
BTC	ORG	0.74	0.09	0.16	238.62	0.78	0.35	0.48	12.09	201.08%	250.71
	MISC	0.16	0.44	0.23		0.33	0.62	0.43		83.55%	
	LOC	0.63	0.65	0.64		0.73	0.69	0.71		10.87%	
	PER	0.74	0.56	0.64		0.83	0.61	0.7		10.3%	

[59] with a fixed learning rate of 0.0015 and batch size of 32. We compute the macro-F1 score after each training epoch on the validation set, and select the best checkpoint to compute the performance score on the test set. Here, we also enforce early stopping after 20 continuous epochs.

The training details of Phrase Embedder with two different loss functions that we explore are compiled in Table II. We also provide the validation performance of Entity Classifier that we train, in each case, after the Phrase Embedder. We achieve a better performing Entity Classifier when the Phrase Embedder is trained with Triplet loss. So we set our implementation to this variant of Phrase Embedder and Entity Classifier when reporting performance in our subsequent analyses.

A. *Evaluating NER Globalizer*

We evaluate the effectiveness of NER Globalizer on its primary objective of using collective processing for better NER performance. To this end, we compute the performance boost brought by Global NER by comparing F1-score at the end of our system's Local and Global NER steps for multiple datasets and check for improvement. We also compare the performance of NER Globalizer to state-of-the-art Local NER systems that, although effective, follow conventional processing, as well as, other Global NER baselines that have their own design of mining global information for good NER performance.

Comparison with Local NER Systems: In Table III we show the performance of state-of-the-art but conventional Local NER techniques for four pre-set entity types on our evaluation datasets. We use Aguilar et al. [3] and also fine-tune a BERT encoder (BERT-NER) that is the current state-of-the-art for NER. NER Globalizer is able to significantly outperform these strong baselines on both streaming and non-streaming datasets.

TABLE V: Effectiveness of Global NER systems

Dataset	Global NER System	Entity Type (F1 score)				Macro F1
		PER	LOC	ORG	MISC	
D_1	NER Globalizer	0.84	0.87	0.5	0.39	**0.65**
	HIRE-NER	0.48	0.5	0.12	0.13	0.31
	DocL-NER	0.4	0.77	0.28	0.39	0.46
	Akbik et al.	0.4	0.47	0.29	0.44	0.4
D_2	NER Globalizer	0.82	0.68	0.54	0.58	**0.66**
	HIRE-NER	0.41	0.62	0.17	0.16	0.34
	DocL-NER	0.53	0.67	0.28	0.37	0.46
	Akbik et al.	0.44	0.62	0.49	0.33	0.47
D_3	NER Globalizer	0.9	0.81	0.71	0.51	**0.73**
	HIRE-NER	0.69	0.68	0.26	0.32	0.49
	DocL-NER	0.31	0.35	0.39	0.12	0.29
	Akbik et al.	0.7	0.78	0.3	0.38	0.54
D_4	NER Globalizer	0.83	0.91	0.68	0.71	**0.78**
	HIRE-NER	0.42	0.73	0.21	0.16	0.38
	DocL-NER	0.36	0.46	0.12	0.09	0.26
	Akbik et al.	0.39	0.65	0.61	0.33	0.5
WNUT 17	NER Globalizer	0.76	0.71	0.52	0.44	**0.61**
	HIRE-NER	0.48	0.5	0.12	0.13	0.31
	DocL-NER	0.49	0.55	0.10	0.13	0.32
	Akbik et al.	0.59	0.59	0.14	0.17	0.37
BTC	NER Globalizer	0.7	0.71	0.48	0.43	**0.58**
	HIRE-NER	0.48	0.59	0.25	0.11	0.36
	DocL-NER	0.45	0.61	0.33	0.09	0.37
	Akbik et al.	0.49	0.62	0.35	0.10	0.39

Performance improvement with Global NER: We zoom into the potential of Global NER by comparing the improvements it brings upon the local module, BERTweet, which itself is an effective Local NER system. The columns under 'Local NER' in Table IV show the performance of conventional language modeling that is fine-tuned to recognize entities of four pre-set types. BERTweet has reportedly set the performance bar on the WNUT17 dataset thus being an already robust model. Hence we check if our Global NER module can still bring in meaningful improvements. The columns in Table IV under 'Global NER' show the NER performance once the Global NER components have been executed. The column F1

Gain indicates the percentage gain in F1 score achieved from Local to Global NER for each entity type in a dataset. This is the improvement that NER Globalizer achieves on top of the Local NER module. The column Time Overhead notes the additional execution time (in seconds) following Local NER.

These results show that Global NER produces an average Macro-F1 gain of 47.04% across all datasets. The average individual performance gains for the four entity types are: a) 11.49% for Person, b) 22.58% for Location, c) 174.37% for Organization, and d) 173.39% for Miscellaneous. The highly uneven improvements across different entity types is due to BERTweet's under-performance in detecting the less frequent and contextually divergent entity types, Organization and Miscellaneous, and its tendency to label their entity mentions as instances of the other two types. NER Globalizer corrects many of these mislabelings thereby producing close to 2 times F1 improvement for these types with a minuscule overhead in execution time.

Improvement on Streaming Datasets: For datasets D_1-D_4 that retain the inherent properties of Twitter streams, NER Globalizer yields an average Macro F1 gain of 49.89%. For individual entity types, the average F1 gains are: a) 11.72% for Person, b) 20.07% for Location, c) 171.28% for Organization, and d) 229.61% for Miscellaneous.

Improvement on Non-Streaming Datasets: Datasets WNUT17 and BTC are random samplings off the Twitter sphere, avoiding the latter's tendency to repeat the same entities within streams. However, NER Globalizer is still able to improve upon its Local NER performance, albeit to a less significant degree than streaming datasets. In this case, the average Macro F1 gain across all Local EMD systems is 41.35%. For individual entity types, the average F1 gains are: a) 11.03% for Person, b) 27.60% for Location, c) 180.54% for Organization, and d) 60.95% for Miscellaneous.

Comparison with Global NER Baselines: We compare the performance of three Global NER systems: Akbik et al. [7], HIRE-NER [40] and DocL-NER [41], with NER Globalizer on all the annotated datasets in Table V. Here we test how effectively global information is captured in each system. NER Globalizer consistently outperforms other Global NER baselines across all datasets by 47.39% on Macro F1, especially by achieving higher precision. Existing Global NER baselines simultaneously update global features for every unique token encountered during training in their memory structures and append them to local token embeddings to infer final output labels of tokens in a sentence. Adding non-local contextual information inevitably introduces noise which can interfere with the decoder's inference of output labels. Distinct from this, NER Globalizer limits curating global contexts only for entity candidates that are aggregated from the local contexts of a candidate's mentions from within the entire scope of the stream. Using this the system is able to better separate entities from noisy candidates.

B. Ablation Study on Framework Components

While it is evident that the proposed NER framework is capable of enhancing performance upon Local NER, we wanted to take a closer look at how the individual framework components contribute towards the overall NER performance. To this end, we execute the framework at different stages of its execution by incrementally adding components, thereby resulting in two additional baseline variants. Here, we use the entire collection of annotated streaming datasets (D_1-D_4) as the test set. Figure 3 shows the improvement in performance as individual system components are added. From bottom to top, the first curve (with only Local NER) reports the weakest performance, proving the limitations of the standalone local system in capturing all the entity mention variations within the stream. The two middle curves are the NER performance we get just by following up Local NER with first the mention extraction process that simply adds missed mentions of entities detected in the local EMD phase. Surface forms that correspond to multiple entity types are handled by assigning the most frequent type. The next baseline adds the local candidate embedding generation step, where the entity classifier is trained only on local embeddings. The topmost curve is the performance yielded at the end of run of the entire framework with the inclusion of global embeddings. BERTweet [4] is a very competitive NER system. Even for such a system, NER Globalizer is still able to significantly improve on its NER effectiveness over the streaming datasets. Following up its execution with just the candidate mention extraction process gives a modest improvement of 12.32% where the focus is mainly on improving the recall by yielding more consistent mention detection across tweets. Upon using the local embeddings we estimate an average improvement of 29.88% in Macro-F1 score from the local system. However the full potential of NER Globalizer is attained when using the global embeddings yielding an average overall improvement of 49.89% across all streaming datasets. This is because with all components of Global NER in place, candidates are further verified and false positives are removed.

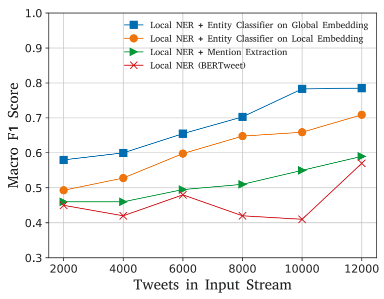


Fig. 3: Impact of Components on Performance

C. Error Analysis

Though NER Globalizer improves upon traditional NER systems, it is not perfect. Here we analyze its errors.

1) If Local NER misses every mention of an entity, its surface form will likely not be added to the CTrie or be considered during Global NER and all its mentions will go undetected by NER Globalizer. Of the 11412 entity mentions

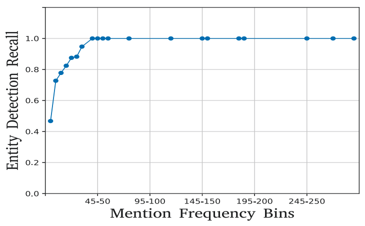


Fig. 4: Impact of Frequency on Detecting Entities

in our streaming datasets from 2306 unique entities, the NER Globalizer failed to find 3008 (26.35%) mentions of 1018 entities that are entirely missed by the local NER module.

2) If Global NER mistypes a candidate, then every mention in its candidate cluster will either be left out as a non-entity or associated with an incorrect type. This would also include mentions that the Local NER correctly found at first. A false negative from Entity Classifier thus hinders the system's objective to correctly recover entity mentions that the Local NER had missed. In our experience, it is rare that an entity mention correctly detected by Local NER is mislabelled at the global step. Of the 11412 entity mentions in our streaming datasets, NER Globalizer mistypes only 1093 mentions (9.57%) due to error by the Entity Classifier.

3) *Handling of long-tailed entities:* To better understand the false negatives from Global NER, we take a look at how the Entity Classifier's performance changes as more mentions of an entity are found in a stream. Figure 4 shows that it is consistently able to correctly detect high-frequency entities from the streaming datasets. We group the annotated entities of different mention frequency in bins of width 5 and track the classifier's recall in labelling them correctly. For infrequent entities, the recall is modest – around 46.8% for entities with 5 or less mentions. But it increases quickly with mention frequency and most frequent entities are correctly labelled. This validates the intuition that collectively processing more mention variations of an entity leads to more robust global embeddings, avoiding randomness. Although long-tailed entities are a common issue in NER, our system can still rectify the mislabeling of many such entities by collecting more instances further downstream.

D. Discussions

Here we summarize some additional implications we observed upon evaluating the proposed system:

- **EMD Gains**: Although the focus of this framework is to solve the NER problem within a collective processing setup, the enhancements we make here also have improved the EMD performance, especially when compared to its previous collective processing counterpart EMD Globalizer. The average improvement in EMD F1 score is 7.9% across all our annotated datasets. The gains in EMD performance come from the NER framework better handling surface form ambiguity, especially by maintaining high precision in cases where an entity shares the same surface form as a non-entity.
- **Handling surface form ambiguity**: A challenge with NER is when entity types share the same surface form with each other or a non-entity. NER Globalizer was able to consistently handle various instances of such ambiguity, e.g., differentiating mentions of the country 'US' from the pronoun us, or 'Fireflies' the song from the insect species.
- **Recall vs Precision Gains**: Our recall gains are consistent across all datasets over all four entity types. We attribute this to the Global NER process that gathers all mentions of candidate surface forms and then groups them to appropriate candidate clusters such that upon validation of their legitimacy by the Entity Classifier all mentions within an entity cluster is produced in the final outcome. Largely we find the curation of global entity embeddings during Global NER to be much more robust to combat local contextual noisiness and incorrect entity classification, thereby yielding considerable improvements in Precision as well. The very occasional drop in Precision (for type ORG with datasets D_1 and D_4 in Table IV) is due to contextual ambiguity, especially for entities with low mention support.
- **Improvement for less popular/mixed-genre entity types**: A distinct trend with NER Globalizer is its strong improvement over the Local NER module for Organization and Miscellaneous types that are less frequent in training sets or cover entities of multiple fine-grained types (like Creative-work/Product). Local NER's predisposition to map entity mentions of these types to more frequent entity types like Person/Location is often due to confounding local context that collective processing can overcome. On average, we saw an F1 Gain of about 1.75 times for Organization and Miscellaneous types. We link the occasional drop in Precision in some cases for the 'ORG' class to mistyping stemming from contextual ambiguity, especially among entities with too low mention support to override ambiguous contexts.

VII. Conclusion

In this paper we presented the NER Globalizer pipeline for microblog streams that builds upon the notion of collective processing for better entity extraction. We started with the conventional wisdom for NER to extract candidate surface forms from the local context of individual sentences. But the inadequacy and noisiness of local context make this unsuitable for producing final NER outputs. We then included a Global NER module. It first separates individual mentions within the stream into candidate clusters and uses their local embedding to collectively represent entity candidates. Using these global representations leads to more accurate entity detection and better overall NER performance. NER Globalizer reported superior performance than several state-of-the-art Local and Global NER systems, outperforming the best Global NER baseline by 47.39% in Macro-F1 on average. We also recorded a Macro-F1 improvement of 47.04% from the BERTweet model at the Local NER stage, where traditional NER systems typically cease execution.

Acknowledgements: This work was supported in part by U.S. National Science Foundation 1546480, 1546441, and 1838145 grants, and a gift from NVIDIA Corporation.

References

[1] N. A. Ghani, S. Hamid, I. A. T. Hashem, and E. Ahmed, "Social media big data analytics: A survey," *Computers in Human Behavior*, vol. 101, pp. 417–428, 2019.

[2] L. Derczynski, E. Nichols, M. van Erp, and N. Limsopatham, "Results of the wnut2017 shared task on novel and emerging entity recognition," in *WNUT*, 2017, pp. 140–147.

[3] G. Aguilar, S. Maharjan, A. P. López-Monroy, and T. Solorio, "A multi-task approach for named entity recognition in social media data," in *W-NUT*. ACL, 2017, pp. 148–153.

[4] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "BERTweet: A pre-trained language model for English Tweets," in *EMNLP*, 2020, pp. 9–14.

[5] A. Aljebreen, W. Meng, and E. C. Dragut, "Segmentation of tweets with urls and its applications to sentiment analysis," in *AAAI*, 2021, pp. 12 480–12 488.

[6] L. He, C. Han, A. Mukherjee, Z. Obradovic, and E. C. Dragut, "On the dynamics of user engagement in news comment media," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 10, no. 1, 2020.

[7] A. Akbik, T. Bergmann, and R. Vollgraf, "Pooled contextualized embeddings for named entity recognition," in *NAACL-HLT*, 2019, pp. 724–728.

[8] S. Saha Bhowmick, E. C. Dragut, and W. Meng, "Boosting entity mention detection for targetted twitter streams with global contextual embeddings," *IEEE International Conference on Data Engineering (ICDE)*, pp. 1085–1097, 2022.

[9] L. Derczynski, K. Bontcheva, and I. Roberts, "Broad Twitter corpus: A diverse named entity recognition resource," in *COLING*, 2016, pp. 1169–1179.

[10] L. A. Ramshaw and M. P. Marcus, "Text chunking using transformation-based learning," in *Natural language processing using very large corpora*. Springer, 1999, pp. 157–176.

[11] A. Ritter, S. Clark, Mausam, and O. Etzioni, "Named entity recognition in tweets: An experimental study," in *EMNLP*, 2011, pp. 1524–1534.

[12] K. Bontcheva, L. Derczynski, A. Funk, M. A. Greenwood, D. Maynard, and N. Aswani, "TwitIE: An open-source information extraction pipeline for microblog text," in *RANLP*, 2013, p. 83–90.

[13] X. Liu, S. Zhang, F. Wei, and M. Zhou, "Recognizing named entities in tweets," in *HLT*, 2011, p. 359–367.

[14] C. Cherry and H. Guo, "The unreasonable effectiveness of word representations for twitter named entity recognition," in *HLT*, 2015, p. 735–745.

[15] E.-S. Yang and Y.-S. Kim, "Hallym: Named entity recognition on twitter with word representation," in *Proceedings of the Workshop on Noisy User-generated Text*, 2015, pp. 72–77.

[16] Z. Toh, B. Chen, and J. Su, "Improving twitter named entity recognition using word representations," in *WNUT*, 2015, p. 141–145.

[17] E.-S. Yang, Y.-B. Kim, R. Sarikaya, and Y.-S. Kim, "Drop-out conditional random fields for twitter with huge mined gazetteer." in *HLT-NAACL*, 2016, pp. 282–288.

[18] M. S. Akhtar, U. K. Sikdar, and A. Ekbal, "Iitp: Multiobjective differential evolution based twitter named entity recognition," in *W-NUT*, 2015, pp. 61–67.

[19] T. Tian, M. Dinarelli, and I. Tellier, "Lattice: Data adaptation for named entity recognition on tweets with features-rich crf," in *ACL 2015 workshop WNUT*, 2015, pp. 68–71.

[20] S. Mishra and J. Diesner, "Semi-supervised named entity recognition in noisy-text," in *WNUT*, 2016, pp. 203–212.

[21] P. von Däniken and M. Cieliebak, "Transfer learning and sentence level features for named entity recognition on tweets," in *WNUT*, 2017, p. 166–171.

[22] F. Dugas and E. Nichols, "Deepnnner: Applying blstm-cnns and extended lexicons to named entity recognition in tweets," in *WNUT*, 2016, p. 178–187.

[23] Q. Zhang, J. Fu, X. Liu, and X. Huang, "Adaptive co-attention network for named entity recognition in tweets." in *AAAI*, 2018.

[24] F. Dernoncourt, J. Y. Lee, and P. Szolovits, "NeuroNER: an easy-to-use program for named-entity recognition based on neural networks," 2017.

[25] M. N. Gerguis, C. Salama, and M. W. El-Kharashi, "Asu: An experimental study on applying deep learning in twitter named entity recognition," *WNUT*, p. 188–196, 2016.

[26] Y. Shen, H. Yun, Z. C. Lipton, Y. Kronrod, and A. Anandkumar, "Deep active learning for named entity recognition," *arXiv preprint arXiv:1707.05928*, p. 252–256, 2017.

[27] S. Zhang, L. He, E. C. Dragut, and S. Vucetic, "How to invest my time: Lessons from human-in-the-loop entity extraction," in *SIGKDD*, 2019, pp. 2305–2313.

[28] S. Zhang, L. He, S. Vucetic, and E. C. Dragut, "Regular expression guided entity mention mining from noisy web data," in *EMNLP*, pp. 1991–2000.

[29] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," in *ACL*, 2016, pp. 1064–1074.

[30] R. Panchendrarajan and A. Amaresan, "Bidirectional lstm-crf for named entity recognition," in *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, 2018.

[31] B. Strauss, B. Toma, A. Ritter, M.-C. de Marneffe, and W. Xu, "Results of the wnut16 named entity recognition shared task," in *WNUT*, 2016.

[32] B. Y. Lin and W. Lu, "Neural adaptation layers for cross-domain named entity recognition," *arXiv preprint arXiv:1810.06368*, 2018.

[33] N. Peng and M. Dredze, "Multi-task domain adaptation for sequence tagging," *arXiv preprint arXiv:1608.02689*, 2016.

[34] C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B.-S. Lee, "Twiner: Named entity recognition in targeted twitter stream," in *SIGIR*, 2012, pp. 721–730.

[35] K. Wang, C. Thrasher, E. Viegas, X. Li, and B.-j. P. Hsu, "An overview of microsoft web n-gram corpus and applications," in *NAACL HLT 2010 DEMO*, 2010, pp. 45–48.

[36] X. Liu, M. Zhou, F. Wei, Z. Fu, and X. Zhou, "Joint inference of named entity recognition and normalization for tweets," in *ACL*, 2012, p. 526–535.

[37] M. B. Habib and M. van Keulen, "Need4tweet: a twitterbot for tweets named entity extraction and disambiguation," p. 31–36, 2015.

[38] I. Yamada, H. Takeda, and Y. Takefuji, "Enhancing named entity recognition in twitter messages using entity linking," in *WNUT*, 2015, p. 136–140.

[39] A. Gattani, D. S. Lamba, N. Garera, M. Tiwari, X. Chai, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan, "Entity extraction, linking, classification, and tagging for social media: a wikipedia-based approach," *PVLDB*, vol. 6, no. 11, pp. 1126–1137, 2013.

[40] Y. Luo, F. Xiao, and H. Zhao, "Hierarchical contextualized representation for named entity recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 8441–8448.

[41] T. Gui, J. Ye, Q. Zhang, Y. Zhou, Y. Gong, and X. Huang, "Leveraging document-level label consistency for named entity recognition." in *IJCAI*, 2020, pp. 3976–3982.

[42] S. Saha Bhowmick, E. C. Dragut, and W. Meng, "Twics: Lightweight entity mention detection in targeted twitter streams," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[44] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186.

[45] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[46] A. Conneau and D. Kiela, "Senteval: An evaluation toolkit for universal sentence representations," *arXiv preprint arXiv:1803.05449*, 2018.

[47] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation," *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017.

[48] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 661–18 673, 2020.

[49] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *EMNLP-IJCNLP*. Association for Computational Linguistics, 2019, pp. 3982–3992.

[50] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 297–304.

[51] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification." *Journal of machine learning research*, vol. 10, no. 2, p. 207–244, 2009.

[52] N. Frosst, N. Papernot, and G. Hinton, "Analyzing and improving representations with the soft nearest neighbor loss," in *International conference on machine learning*. PMLR, 2019, pp. 2012–2020.

[53] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*. PMLR, 2007, pp. 412–419.

[54] K. C. Gowda and G. Krishna, "Agglomerative clustering using the concept of mutual nearest neighbourhood," *Pattern recognition*, vol. 10, no. 2, pp. 105–112, 1978.

[55] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5147–5156.

[56] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep adaptive image clustering," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5879–5887.

[57] A. Hadifar, L. Sterckx, T. Demeester, and C. Develder, "A self-training approach for short text clustering," in *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, 2019, pp. 194–199.

[58] K. Lee, D. Palsetia, R. Narayanan, M. M. A. Patwary, A. Agrawal, and A. Choudhary, "Twitter trending topic classification," in *ICDM*, 2011, pp. 251–258.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.