University of Pennsylvania LoReHLT 2019 Submission

Stephen Mayhew[1], Tatiana Tsygankova[1], Francesca Marini[1], Zihan Wang[5], Jane Lee[1],
Xiaodong Yu[1], Xingyu Fu[5], Weijia Shi[2], Zian Zhao[3],
Wenpeng Yin[1], Karthikeyan K[1,4], Jamaal Hay[1],
Michael Shur[1], Jennifer Sheffield[1], Dan Roth[1]
[1]*University of Pennsylvania, Department of Computer and Information Science, Philadelphia, Pennsylvania*
[2]*University of California, Los Angeles, Department of Computer Science, Los Angeles, California*
[3]*Columbia University, Department of Computer and Information Science, New York, New York*
[4]*Indian Institute of Technology, Kanpur, Department of Computer Science and Engineering*
[5]*University of Illinois at Urbana-Champaign, Computer Science Department, Urbana, Illinois*
{mayhew, ttasya, jamaalh, xdyu, wenpeng, kkarthi, shurm, sheffiel, danroth}@seas.upenn.edu,
{janehlee, fmarini}@sas.upenn.edu, swj0419@g.ucla.edu, zhao.zian@columbia.edu, {zihanw2, xingyuf2}@illinois.edu

***Abstract*—This document presents the system description of the University of Pennsylvania team's submission to the LoReHLT2019 task. It describes the details of our submissions in the Entity Discovery and Linking (EDL) and Situation Frame (SF) tasks. Our systems achieved top-ranked scores of 79.4 (IL11) and 79.5 (IL12) for the NER task on the IL data, and top-ranked score of 56.6 (IL11) for EDL on the IL data, and top-ranked score of 37.80 (IL11) for SF typing on the IL data[1].**

***Index Terms*—Named Entity Recognition, Entity Discovery and Linking, Situation Frame, Natural Language Processing, Low-Resource Languages**

I. Introduction

This document gives details on the University of Pennsylvania's participation in the Low-Resource Human Language Technologies (LoReHLT) evaluation of 2019, put on by the National Institute for Standards and Technology (NIST) and supported by the Linguistic Data Consortium (LDC). This year is the fourth and final running of the evaluation, and our fourth year of participation (although second at the University of Pennsylvania).

In this evaluation, participants address certain natural language processing (NLP) tasks in the context of little or no training data, a situation commonly referred to as low-resource NLP. In the weeks and months leading up to the evaluation, participants prepare methods and systems for rapidly developing NLP tools for low-resource languages. When the evaluation itself begins, participants are given unannotated data in previously unknown "surprise" languages, and tasked with developing tools for these languages within a short timeframe.

The first year, 2016, had a single surprise language, Uyghur, and three tasks: Named Entity Recognition (NER), Situation Frame (SF), and Machine Translation (MT). We participated only in the NER and SF tasks, and documented our submissions in [13]. The second year, 2017, expanded to two surprise languages (Tigrinya and Oromo) and extended NER to Entity Discovery and Linking (EDL) while keeping the other tasks the same (except for some small changes to the SF task definition). Our submissions to the EDL and SF tasks are described in [9]. In the third year, the tasks remained mostly the same as the prior year, and the surprise languages (or 'Incident Languages' or 'ILs') were Kinyarwanda (IL9) and Sinhalese (IL10).

This year, the tasks remained the same, and the incident languages (ILs) were Odia (IL11) and Ilocano (IL12).

The format of the evaluation meant that we had to provide results after 24 hours (checkpoint 1), and then after an additional 8 days (checkpoint 2). Below we give a brief introduction to each language.

a) Odia (IL11): Odia, formerly known as Oriya, is an Indo-Aryan language natively spoken by 38 million people in the Indian state of Odisha and some neighboring states.[2] It is one of the many official languages of India, and has a history of borrowing words from Telugu and Tamil. The Odia script is a member of the Brahmic script family, developed from the Kalinga script, but is unique to Odia and sometimes used as a regional writing system for Sanskrit.

b) Ilocano (IL12): Ilocano is the third most-spoken native language of the Philippines, spoken natively by 9 million people.[3] Ilocano is a regional Austronesian language spoken in the northern part of Luzon and is sometimes referred to as Ilokano, Iloco or Iluko, though some people consider Ilocano as a dialect. It is written in Latin script, and resembles Tagalog. Also like Tagalog, and because of the wide use of English, terms with no local equivalents in Ilocano are often left in English.

[1]SF ranking was released after the submission of system description

[2]https://en.wikipedia.org/wiki/Odia_language

[3]https://en.wikipedia.org/wiki/Ilocano_language

II. Submission Highlights

Before laying out full details of our submissions, we briefly summarize our main approaches in each task.

EDL We separated EDL into NER and entity linking (EL). As in previous years, one of our most important techniques was manual annotation, both from Native Informants (NIs) and also from non-speakers (NSs). One annotation strategy in particular was important: all text that went to an NI was pre-annotated by an NS. We also experimented with different ways of training BERT [2], including training on target language only, training on target language and related languages, and continuing training from the official mBERT model.

The highlights for the EL system were Google-centered candidate generation and classifier plus BERT-centered candidate ranking. Unlike past years, our system employed the Google query log and geographical information for the candidate generation process, while also considering transliterating low-resource language mentions into a high-resource language. A newly trained classifier and a BERT model were utilized for ranking the generated candidates, and proved beneficial for efficiently employing contextual information.

SF We approached the SF task from two angles: as classification, and as textual entailment. In both cases, we only trained our models on English data.

As classification, we implemented rather standard deep neural classifiers (e.g., convolutional neural network) and multilingual BERT. Convolutional classifiers rely on bilingual embeddings, and multilingual BERT works on English and ILs directly. We combined these two types of classifiers.

For the entailment approach, we converted the SF types into hypotheses. We converted our representation learner to an entailment architecture, and we also explored multilingual BERT for entailment.

III. Multilingual Contextual Embeddings

Since both EDL and SF relied on the development of multilingual contextual embeddings, we describe those experiments first in this section before outlining the tasks separately.

Following the tremendous success of contextual word embeddings in the NLP literature, we included several versions of BERT [2] in our experiments. We trained BERT using the code provided by the authors,[4] and with Tensor Processing Units (TPUs) on the Google Cloud.[5]

There exists a pretrained multilingual version of BERT, trained by the authors of the original paper on 104 languages in Wikipedia. We refer to this model as multilingual BERT, or mBERT. As we had anticipated prior to the evaluation, neither of the ILs is present in mBERT, although similar languages are, such as Bengali and Hindi (similar to IL11) and Indonesian and Tagalog (similar to IL12).

The presence of these similar languages opens an avenue for zero-shot learning by training on related languages, but a particular issue with IL11 is that the script of the language is not present in the mBERT vocabulary. This means that nearly every token in IL11 text would be treated as unknown (UNK).

We experimented with several different methods of training BERT that ultimately coalesced into two strategies:

1) BERT trained on the ILs and some related languages, which we refer to as few-BERT, or fBERT
2) mBERT with additional training epochs using the target languages, which we refer to as continued-BERT, or cBERT.

A. Few-BERT (fBERT)

For IL11, we trained from scratch a BERT model using English, the target language (Odia), Bengali, Tamil and Hindi. Since some of the languages, such as Bengali and English, are much higher-resource than Odia, we had to super-sample the low-resource languages and sub-sample the high-resource languages. This gave a much more balanced dataset which we then used to pretrain BERT.

For IL12, a similar method to that described above was used, with the following language set: English, Ilocano, Tagalog, Indonesian, and Malay.

We also trained an fBERT model containing all of the LRLP data (approximately 20 different languages) as well as the language data from the model described above used to train IL11 and IL12.

B. Continued-BERT (cBERT)

Google's mBERT was trained over a large number of languages, and has shown strong performance in cross-lingual tasks [16]. Given this strong performance, and the fact that neither Odia nor Ilocano was present in the pretraining, it could be useful to use mBERT as a starting point for continuing training in the ILs. As the existing mBERT vocabulary does not contain proper word-pieces of Odia or Ilocano (significantly, it does not even contain Odia characters) we can't directly continue pretraining mBERT. Hence, we added new word-pieces to the vocabulary. The new vocabulary contains both Google's vocab and newly added word-pieces.

BERT has encoder and decoder weights whose sizes are proportional to the size of the vocabulary. When we add new word-pieces to the vocabulary, the size changed. As a result, we needed to change the sizes of these matrices, so we couldn't initialize these matrices directly from mBERT. We initialized all other weights directly from mBERT.

The encoder is a $V \times 768$ dimensional matrix, where V is the size of the vocabulary. Note that the i^{th} row of the encoder corresponds to the i^{th} word-piece; hence we initialize all the rows that correspond to an mBERT word-piece with its corresponding mBERT weights, and for new word-pieces we initialize randomly. Further, BERT uses weight tying; i.e., encoder and decoder share the same weight, but the decoder has an extra bias term of dimension $V \times 1$. which is initialized similar to the encoder weights as described above.

We continued pre-training with the above initialized weights in two ways, one with just the IL and the other with the IL and related languages (For IL11 we used Odia, Bengali, Hindi and

[4]https://github.com/google-research/bert/

[5]cloud.google.com

English, and for IL12 we used Ilocano, Tagalog, Indonesian and English). We found that the one that we continued training on IL with related languages performed better in most cases.

IV. Entity Discovery and Linking

We approached EDL in two stages: first doing named entity recognition (NER) in order to find mentions, then doing entity linking (EL) to link these mentions to a knowledge base.

A. NER

Our work in NER followed two main threads: preparation of data in the target languages, and development of strong cross-lingual models.

1) Data Preparation: The first step in building a low-resource NER system is to gather data in the target language. This process took several steps: Document Selection, Development Sets, and Manual Annotation.

Document Selection Since the first step in the process of NER involves the manual annotation of documents, we hoped to select documents for both the development and training sets which were similarly distributed to set E. We know that for each incident language, set E contains documents about specific incidents, so we hoped to devise a way of selecting documents that would either be centered around the incidents themselves or else be of a similar type, and therefore useful to our models.

We began by constructing lists of vocabulary words in the ILs that would be potentially relevant to the regions where the ILs are spoken, in this case Odisha, India (where Odia is spoken) and Northern Luzon, Philippines (where Ilocano is spoken). These lists included the names of important geopolitical entities (such as countries, provinces, and cities), politicians (such as the name of the president), and international organizations (such as the UN), as well as disaster-related words that might occur more frequently in incident-specific documents (such as *emergency*, *fuel*, *cyclone*, or *protest*). We could then prioritize the selection of documents which contained at least one of the words on the vocabulary lists.

We also sought to keep our sets fairly proportional to set E in regards to the document types included. For example, if 25% of the tokens in set E came from newswire, then we tried to select approximately 25% of the tokens for our development and train sets from the body of newswire documents available to us for selection, using the vocabulary lists as the means of selecting documents from each document type.

We evaluated this data selection method by selecting documents from the train split of the Ontonotes data set [15]. We used the Ontonotes gold annotated data as our test set. We compared the performance of this "proportional selection by vocabulary list" method to both random selection and querying for documents similar to the test set based on TF-IDF, and we found that our method worked best. We also tested our method using the Sinhalese and Kinyarwanda data from the 2018 LoReHLT evaluation and found similar results.

Table I
Numbers for manually annotated datasets in IL11. # Entities refers to the number of name phrases (not necessarily unique) in the data. For example, "Bernard Ver" is one name phrase.

Dataset		Source	# Docs	# Toks	# Sents	# Ents
All		Sets 0/1	474	57,636	4,057	3,711
NI	Extra	Sets 0/1	53	10,203	813	839
	Dev0	Set 0	35	3,591	255	234
	Dev1	Set 1	32	13,571	810	853
NS	Extra	Sets 0/1	81	26,337	1,906	1,461
	Tweets	Set 0	273	3,934	273	322

Building Development Sets Since our data selection method is dictated by the ILs themselves and constrained by the time it takes to assemble sufficient lists of important vocabulary words in those languages, we were unable to use this method to select documents for Development set 0 (Dev0) and Train set 0 (Train0). For these sets, we selected the data randomly from set 0 because we could not afford the time between the first and second checkpoints to assemble the vocabulary lists. For our Development and Train sets from both set 0 and set 1, we did not select any of the RF (reference) documents, as these did not appear within set E, and would likely be unhelpful.

We had scheduled our first Native Informant (NI) block for the end the first day, with the intention of having them annotate Dev0 as fully as possible. In preparation for the NI annotations, we annotated as much of Dev0 as we could manually. We then provided the NI with our annotations and instructed them to add or fix any incorrect annotations that they saw. This process ran smoothly, and we believe that the NI was able to annotate more efficiently because we had already provided a rudimentary annotation of the documents.

As checkpoint 2 approached, we had already completed constructing the vocabulary lists, so we could assemble the next Development set (Dev1) and Train set (Train1) as soon as we could access the post-incident set 1. Once these were selected, we annotated as much as possible throughout the course of the evaluation. During the NI sessions, we asked the NIs to fix our pre-existing annotations within Dev1 (and later within the train sets) proceeding on the assumption that these "NI-checked" documents would be as close to gold data as we could get.

Note that when we remark on performance in the following sections, we are referring to performance on our noisy Development sets, and not on gold data. As of writing, we do not have access to any gold data in the ILs, but only results from the evaluation submissions. In some cases, we can compare development results against official results, but not in all.

Manual Annotation The first step of our approach was to annotate data manually, using Native Informants as time permitted to fix the annotations we had already made. For IL11, we had 5 hours of NI time, with the first hour available for checkpoint 1 and the remainder for checkpoint 2. For IL12,

Table II
NUMBERS FOR MANUALLY ANNOTATED DATASETS IN IL12.

Dataset		Source	# Docs	# Toks	# Sents	# Ents
All		Sets 0/1	325	98,331	5,463	4,847
NI	Extra	Sets 0/1	148	45,096	2,619	2,528
	Dev0	Set 0	28	24,086	1,523	575
	Dev1	Set 1	75	16,452	699	1,129
NS	Extra	Sets 0/1	28	12,050	576	564
	Tweets	Set 0	46	647	46	40

we had 6 hours of NI time, with the first hour available for checkpoint 1, and the remainder for checkpoint 2. We obtained the 6^{th} hour for IL12 by using one of the SF NI sessions for NER annotation. For all annotation, we used TALEN [8], with romanization done either with the unidecode Java library[6] or Uroman [5].

As described above, we used the checkpoint 1 hour to get our Dev0 annotations fixed, and then the next hour to fix our Dev1 annotations. This left the remaining NI hours for each language for further fixing of annotations.

Before the NIs fixed our annotations of the Development sets we had split the sets of approximately 20K tokens into 5 disjoint subgroups (we ended up splitting Dev1 of IL11 into 10 subgroups), which people could sign up to annotate. Beyond the Development sets, our manual annotation process was the following. For Train0 of both ILs, we had intended to select 100K tokens from set 1 of both IL11 and IL12; however, the entirety of set 1 for IL12 was less than 100K tokens. As a result, we ended up selecting the rest of set 1 which had not been used to make Dev1 as Train1. We then selectively removed certain documents which appeared to be less relevant (such as recipes or the table of contents for magazines). We split this Train1 into about 20 subgroups. For IL11, we used the document selection method described above to select a corpus of 100K words from Set 1. We split this Train1 into about 40 subgroups. We created a sign-up sheet for each group so that members of our team could sign up for subgroups to annotate when they had spare time.

We wished to include Tweets in our training data since there are Tweets in Set E for both ILs. However, neither set 1 contained any Tweets, so we took them from set 0. We spent some time toward the end of the evaluation manually annotating these Tweets, and we also wrote some annotation rules for post-processing that tag Twitter hashtags and handles as named entities when appropriate.

In the interest of gathering the broadest variety of annotations, we never duplicated annotations. As a result, we cannot measure inter-annotator agreement or adjudicate disagreements. However, we did have our most experienced annotator review all of the non-speaker annotations before the training of the final models to ensure that guideline interpretations were consistent among all annotators. This proved to be very useful practice since the quality of the annotations (evaluated on our Dev1 set) improved greatly.

As we were annotating text, certain ambiguities forced us to interpret the annotation guidelines. We realize that our interpretations could be incorrect, so we record here some of the more important decisions we made.

- For both IL11 and IL12, we consistently chose to not annotate the generic term 'police' unless it was referenced in the document as an established organization, for example, "Philippine National Police", as per the annotation guidelines.

- For both IL11 and IL12, we noticed that a number of documents in set0 contained references to religious figures (seemingly citing Bible passages), which we refrained from tagging based on the clarifications on this subject in the annotation guidelines.

- For both IL11 and IL12, we included the "@" sign in the span of a Twitter handle named entity, even if there was a space separating them, to maintain consistency with the other, untokenized, Twitter handle examples.

- For IL11, we also found many posts by the Twitter account "PMO India" (Office of the Prime Minister of India, @PMOIndia), which we tagged as an ORG.

- For IL12, we noticed that many of the documents in set 1 included the same closing paragraph, most likely citing the source of the document as "Bannawag Magazine" and listing ways to access this content in other domains. For the majority we chose to classify all instances of "Bannawag" as ORG, although for one submission we excluded all annotations of "Bannawag" since we never noticed this entity referred to outside of the subtext at the bottom of the documents.

- For IL12, we also chose to include the terms 'siudad' (city) and 'brgy.' (province) within the span of the GPE entity it is referring to, based on a similar judgment call in Tagalog annotations.

The final statistics of all annotated data can be found in Tables I and II for IL11 and IL12 respectively. No members of our team had any experience with either language.

Post-processing As a final step for all submissions, we used a rule-based system to post-process predictions made by our NER model. In general, the rules we used were either generated from the predictions of the model (to ensure that all instances of the same entity were recognized) or computed manually, using popular Twitter hashtags and the respective IL knowledge base as reference. While there was some variation in terms of which rules were applied to which submissions, a simple rule tagging all missed Twitter handles (tokens starting with an '@' symbol) as PER entities was used uniformly for all of the submissions.

[6]github.com/jirutka/unidecode

Table III
SCORES ON DEV1 WITH DIFFERENT EMBEDDINGS FOR IL11. BOTH MODELS ARE TRAINED ON ALL NI ANNOTATIONS.

Method	P	R	F1
fastText	54.0	34.3	42.0
fBERT	67.1	60.2	63.5

Table IV
SCORES ON DEV1 WITH DIFFERENT EMBEDDINGS FOR IL12. BOTH MODELS ARE TRAINED ON ALL NI ANNOTATIONS AND NS ANNOTATIONS.

Method	P	R	F1
fastText	78.9	77.4	78.2
fBERT	86.0	80.0	82.8

2) NER Model: We selected LSTM-CNNs-CRF [7] as our NER model. We mainly focused on two parts, *data selection* and *pretrained embeddings*, to improve the performance of our NER model. We used the python library AllenNLP [4] to train our models, using almost all parameters from the default NER config.

Data Selection Our training data on the IL is made up of NI and NS annotated data, introduced in Section IV-A1. We also extended our training data by including NER data from similar languages from the provided LORELEI Language Resource Packs (LRLPs) [11]. For IL11 (Odia), we chose Bengali[7] and Hindi[8]; for IL12 (Ilocano), Tagalog[9] and Indonesian[10]. For both languages, we also used English NER data, including the LORELEI language pack[11], and a re-annotation of about half of the CONLL2003 English data [12] to remove MISC tags and include GPE tags. We noticed that, for both languages, including extra data brought improvements on our development set. We also tried some experiments using lemmatized versions of the text, which seemed not to improve model performance, and therefore used the original data in training all submitted models.

Pretrained embeddings Pretraining word embeddings have been useful for NER. We experimented with

- Fasttext [1], a non-contextual word embedding model that incorporates subword information when learning word representations. We trained fastText on each IL, with the provided IL data, and text data from Wikipedia that we gathered before the evaluation.
- BERT [2], a language model based on based on transformers (see Section III).

Experiments

The results in Tables III and IV show that fBERT outperforms fastText on both IL11 and IL12.

[7]LDC2017E60
[8]LDC2017E62
[9]LDC2017E68
[10]LDC2017E66
[11]LDC2019E01

Table V
SCORES ON DEV1 WITH FBERT FOR IL11

Data	P	R	F1
IL (NI)	67.1	60.2	63.5
Bengali	55.5	62.9	59.0
Hindi	52.8	62.0	57.1
Bengali + Hindi	61.4	70.2	65.5
IL (NI) + Bengali + Hindi	66.0	71.8	68.8
IL (NI) + Bengali + Hindi + machine	69.0	74.2	71.5

Table VI
SCORES ON DEV1 WITH FBERT FOR IL12

Data	P	R	F1
IL (NI+NS)	86.0	80.0	82.8
Tagalog	65.5	74.3	69.6
Indonesian	61.5	71.1	66.0
Tagalog + Indonesian	71.8	75.7	73.7
IL (NI+NS) + Tagalog + Indonesian	82.2	83.7	83.0
IL (NI+NS) + Tagalog + Indonesian + machine	80.4	83.9	82.1

We noticed that because of the multilingual effectiveness of BERT, training on related languages and then zero-shot transferring into IL language gave non-trivial performance, see Tables V and VI. We also noticed that incorporating related languages and machine annotations helped in IL11, but did not help or even harmed in IL12. We hypothesize that this difference may come from quality and quantity of IL training data we have in these two languages.

From the results in Tables VII and VIII, we conclude that cBERT performed better than fBERT trained from scratch. Note that for these two tables, we used annotated tweets, fixed NS annotations and English data, so the results are not directly comparable to other tables.

Non-neural Systems In addition to the BiLSTM-CRF models, we also used the Cogcomp NER system [6], [10], a linear model based on averaged perceptron that has been our default model for the past several years. We trained Brown clusters for each language separately, using cluster sizes of 500, 1000, and 2000. When trained on IL text, this model produced modest performance on the dev sets.

Table VII
SCORES ON DEV1 FOR IL11

Method	P	R	F1
fBERT	66.7	78.9	72.3
cBERT	71.2	78.2	74.5

Table VIII
SCORES ON DEV1 FOR IL12

Method	P	R	F1
fBERT	82.6	82.7	82.7
cBERT	83.9	85.2	84.5

When adding other related languages, the performance also increased slightly, despite having Brown clusters only in the target language. We included Cogcomp NER results in two of our submissions: the first as a combination with the best performing neural model (*sub6-ner*, *sub6-ner-fixed*), and the second as a standalone submission (*sub11-ner*).

B. Entity Linking

Given an IL mention w detected by the NER system, our goal is to link it to an entity in the knowledge base provided, or "NIL" if the corresponding entity is not in the knowledge base. The entity linking task in general is conducted in two main steps: candidate generation and candidate ranking. We use K to refer to the Knowledge Base (KB) provided with the Lorelei Incident Language pack in our description below. The system diagram is as presented in Figure 1.

Candidate Generation The candidate generation process aims at finding the possible entities in K for a given mention w. Since w is given in the IL, we need to find the English format of w, and then link back to the knowledge base. During this process, to gain information about w, we use Wikipedia as an intermediate transaction. Therefore, we first link w to the Wikipedia page of its English form, then link back to K.

The whole process begins with surface normalization, since IL mentions can have different surface names due to changes in tenses, adding of prepositions and language habits. We then generate the English format of the mention. One method is utilizing Google query information and Google geographical information in the IL to generate candidate Wikipedia pages for the mention and get English page through links. Another method is to use multiple transliteration models to map the IL mention to English or higher-resource languages first, then generate English Wikipedia pages by putting the new form into Google query and Google map searching. Finally, the English format of this mention is extracted from the English Wikipedia page of the found page, and linked to K via phrase and token level matching.

One exception from using the Wikipedia page exists in the candidate generation part – back pointer situation. We use Google transliteration to generate English-to-IL pairs for entities having types of Person or Organization and for those in Gazetteer data. Then, given an IL mention, we use a backward pointer to directly link the mention back to the knowledge base if the string matches exactly.

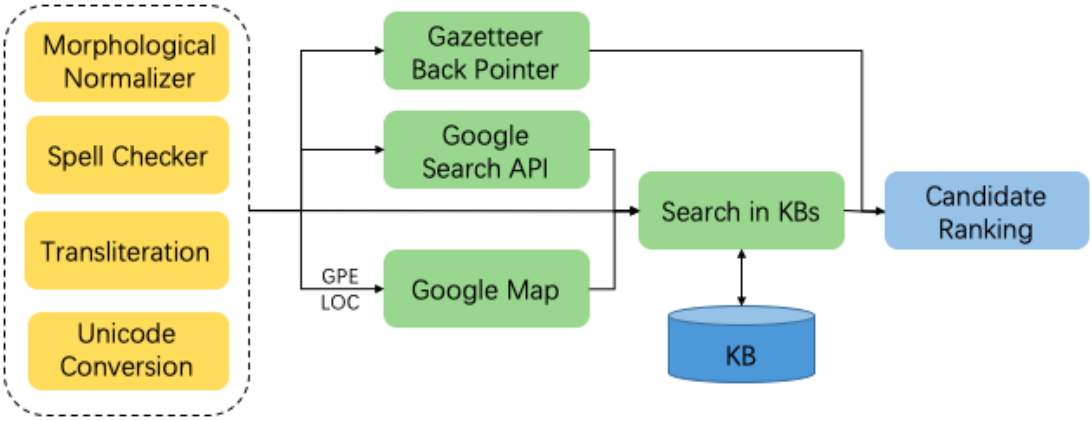
- **Surface Normalization** Variance in word form can make searches based on string similarity difficult. As a result, discovered surfaces are first normalized before candidate generation. We normalized the surfaces using morphological rules and spell-checking via Levenstein distance.
  Each language has its own morphological rules. The system normalizes discovered surfaces using separate rules for IL11 and IL12. For IL12, the Ilocano language has both suffixes and prefixes. However, most of them are applied on verbs, having little effect on named entities that we cared about. So, the system does not apply morphological normalization on the Ilocano language. On the other hand, IL11 has suffixes applied on named entities, which will affect our generation result. A list of suffixes is generated by comparing word forms of discovered entities in set 0 and set 1. The entity linking system then removes possible suffixes of the surface according to the list.
  Another source for different word forms is writing style and spelling errors. In IL11, there exists a special character (U+0b3C) which does not affect the meaning and does appear on certain characters in one of the writing styles. To cope with the issue, the system toggles the character and searches both origin and toggled form during candidate generation. For spelling errors, a frequency-based spell-checker is built on all the entities discovered in set 0 and set 1, which corrects the spelling errors in system input.
- **Transliteration** Transliteration is conducted to provide a larger search base for the surface as there are limited resources for the IL. With the similarity of Indian languages, we use a simple character conversion method to transliterate IL11 to Hindi, a language with richer resources, and conduct candidate generation. In addition to the simple method, a more advanced sequence-to-sequence model using hard attention is used to generate transliterations of strings in English. The training data for such model is collected from filtered Wikipedia title pairs, which is automatically generated using freebase database. All the transliterated surfaces are passed to candidate generation process along with the surfaces introduced in previous paragraph.

Candidate Ranking

To rank the candidate entities we employed the following techniques to assign weights for each candidate. In the system, specific weights adjusted from those features are tuned on the development set.

- **Entity Popularity** To rank which of the entities in K obtained from candidate generation a given mention should link to, we used entity popularity as a feature. Entity popularity was computed using the number of inlinks (in Wikipedia) of the corresponding Wikipedia page, normalized by the sum of inlinks of all candidate entities under consideration.
- **Country Code and Administrative Code** We hypothesize that candidates that are close to the location where the incident takes place will have a higher chance of being the correct entity. For example, Ilocano (IL12) is spoken by people in Ilocos Norte and Ilocos Sur. The mention with the surface form "Banna" written in Ilocano is more likely to refer to the city "Banna" in Ilocos Sur rather than the city in Pakistan with the same name. For every candidate, we assign higher weights to the candidate of which country code and administrative code match the incident location.

Figure 1. EDL System Architecture



- **fBERT** To disambiguate KB entries with external links, we apply fBERT to compute the probability of a mention m linking to entity e using its text context vector g and its Wikipedia context vector w. We first collect a list of entries with the external Wikipedia links among the candidates. Then we average the contextualized word embeddings for the mention in the summary of this Wikipedia page and the word embedding for the mention in the context to compute the cosine similarity between these two vectors. For the candidate with the highest cosine similarity, we assign higher weight to it.
- **Feature Classifier** The exact same location can have different entities associated with the distinct location feature code in Geonames. For example, the GPE entity of "Kigali" has two entries with the feature classes "country" and "city". The way to resolve the ambiguity is to check the context in which the city appears right after "Kigali" in this example. For this issue, we train a classifier to predict the type of the given mention. We use a local context encoder to encode the left and the right context of the mention into two vectors $l \in R^h$ and $r \in R^h$, which are fed into a multilayer perceptron followed by a sigmoid function to calculate the probability of each class.

NIL Clustering Two simple techniques are applied for clustering NIL-linked mentions. (1) **Exact Match** - mentions with identical surface forms are part of the same cluster. (2) **Fuzzy-match** - We first sort mentions in the decreasing order of the number of tokens in their surface. We cluster mentions sequentially from longest to shortest, and add a mention to an existing cluster if there is a $> 50\%$ token overlap between the mention surface and the longest mention in the cluster. Otherwise we generate a new cluster with this mention.

C. Submissions

Our best submissions for both IL11 and IL12 are seen in Table IX, broken down by performance on all data (IL+English), IL data alone, and English data alone. NER scores are *strong_typed_mention_match*, which corresponds to standard F1. EDL scores are *typed_mention_ceaf_plus*. This table shows that we achieved top-scoring performance in the evaluation on NER scores for both IL11 and IL12 at CP2, as well as highest EDL scores for IL11 CP2. Our low English scores reflect the small amount of attention we paid to this task. In the following tables, we report both "All" results and "IL-only" results, with a special focus on the "IL-only" results as being the main goal of the evaluation.

Detailed tables showing our submissions and the resulting scores follow: see Tables X, XI, XII, and XIII. Since the NER results were upstream in the pipeline with regard to EDL, we had many NER results that were never processed with EDL. We have separated our tables to reflect this, showing first NER+EDL results, then NER results only.

For IL11, sub9 (referring to submission 9, with name *il11-sub9-ner*) shows that post-processing with self propagation rules hurts performance significantly (up to 5 points). This was because it damaged the precision by nearly 10 points, but failed to improve recall much. It turns out that when the NER team handed over sub3 results to the EL team, they forgot to post-process the submission, and this became sub3-exp1. Later, a post-processed version of sub3 became sub3-exp2. Given the dramatic difference in performance, we are glad this mistake happened.

In both languages, submissions 19-22 held out the dev1 set in order to tune the model. When comparing against submissions 2-5 (both languages), we see that this harmed performance. This means that the benefit that might have come from choosing a tuned model is less than the benefit from additional training data.

In both languages, comparing sub2 with sub3 and sub19 with sub20, we see that our hypotheses and experiments about the superiority of cBERT over fBERT (as shown in Tables VII and VIII) are borne out.

We experimented with getting more dense data by selecting

Table IX
OUR BEST SUBMISSIONS IL11 AND IL12. "ALL" REFERS TO THE COMBINATION OF ENGLISH AND IL RESULTS. "IL" IS RESULTS ON IL DATA ONLY. "ENG" IS RESULTS ON ENGLISH DATA ONLY. WE CONSIDER THE "IL" SCORES MOST IMPORTANT. NER SCORES ARE *strong_typed_mention_match*, WHICH CORRESPONDS TO STANDARD F1. EDL SCORES ARE *typed_mention_ceaf_plus*. IN THIS TABLE, **BOLD** INDICATES TOP-SCORING SUBMISSION IN THE EVALUATION.

			All		IL		Eng	
Language	CP	Submission Name	NER	EDL	NER	EDL	NER	EDL
IL11	1	il11-exact	48.4	28.8	14.2	8.2	65.8	41.6
	2	il11-sub3-exp1-exact	68.9	44.8	**79.4**	**56.6**	58.1	33.7
IL12	1	mbert_eng	**67.5**	33.1	**70.9**	34.6	**64.7**	38.2
	2	il12-sub3-exp2-exact	61.1	37.3	**79.5**	54.7	45.9	24.6

Table X
IL11 SUBMISSIONS WITH EDL. **BOLD** INDICATES HIGHEST SCORE AMONG OUR SUBMISSIONS.

CP	Submission Name	NER sub used	EDL approach	NER Score	EDL Score
1	il11-exact	fastText (dev0)	Google search and exact matching for nil clustering	**46.4**	**28.8**
1	il11-fuzzy	fastText (dev0)	Google search and fuzzy matching for nil clustering	**46.4**	28.7
2	il11-sub3-exp1-exact	sub3	Google search, transliteration, and type constraint	**68.9**	**44.8**
2	il11-724-exact1	sub3	Google search	**68.9**	44.7
2	il11-724-exact2	sub3	Google search	**68.9**	44.7
2	il11-sub3-exp2-exact	sub3	Google search	66.7	43.4

Table XI
IL11 NER-ONLY SUBMISSIONS. A SMALL NUMBER OF THESE (MARKED WITH *, AND PRESENT IN TABLE X) ALSO HAD EDL PREDICTIONS. ALL TRAIN IS: ALL IL11, SELF-TRAIN 100K, BEN, HIN, ENG. RECALL THAT TWITTER RULES WERE USED AS POST-PROCESSING FOR ALL SUBMISSIONS (UNLESS OTHERWISE NOTED).

CP	Submission Name	Model	Data	Post-processing	NER	NER (IL)
2	il11-sub2-ner	fBERT	All Train	self	70.6	73.4
2	il11-sub3-exp1-exact*	cBERT	All Train	– (no twitter)	68.9	**79.4**
2	il11-sub3-exp2-exact*	cBERT	All Train	self	66.7	74.4
2	il11-sub4-ner	cBERT	Mixed IL data	self	72.3	76.8
2	il11-sub5-ner	cBERT	Dense IL data	self	71.0	74.2
2	il11-sub6-ner	CCG+cBERT	All Train	self	69.7	71.6
2	il11-sub6-ner-fixed	CCG+cBERT	All Train	self	71.0	74.1
2	il11-sub8-ner	cBERT	All Train	entity list + self	71.0	74.2
2	il11-sub9-ner	cBERT	All Train	–	**73.5**	**79.4**
2	il11-sub10-ner	cBERT	All Train	entity list	73.4	79.2
2	il11-sub11-ner	CCG	All Train	self	67.7	67.8
2	il11-sub18-ner	cBERT	Only IL data	self	70.9	74.0
2	il11-sub19-ner	fBERT	All Train (no dev1)	self	70.3	72.8
2	il11-sub20-ner	cBERT	All Train (no dev1)	self	71.0	74.2
2	il11-sub21-ner	cBERT	Mixed IL data (no dev1)	self	70.9	73.9
2	il11-sub22-ner	cBERT	Dense IL data (no dev1)	self	71.2	74.4

Table XII
IL12 SUBMISSIONS WITH EDL

CP	Submission Name	NER sub used	EDL approach	NER Score	EDL Score
1	il12-exact	fastText (dev0)	Google search and exact matching for nil clustering	50.5	27.9
1	il12-fuzzy	fastText (dev0)	Google search and fuzzy matching for nil clustering	50.5	27.2
1	mbert_eng	Eng+Spa+Tgl+Ind	Google search	67.5	**33.1**
1	il12-mbert_eng	Eng+Spa+Tgl+Ind	Google search	**67.7**	21.2
2	il12-sub3-exp1-exact	sub3	Google search, BERT, Classifier	50.1	29.5
2	il12-sub3-exp2-exact	sub3	Google search, BERT, Classifier	**61.1**	**37.3**
2	il12-sub3-exp3-exact	sub3	Google search, BERT, Classifier	**61.1**	**37.3**
2	il12-sub3-exp3-fuzzy	sub3	Google search, BERT	**61.1**	36.5
2	il12-sub5-exp1-exact	sub5	Google search, BERT	60.1	37.1
2	il12-sub5-exp1-fuzzy	sub5	Google search, BERT	60.1	36.2
2	il12-sub5-exp2-exact	sub5	Google search, BERT, Classifier	60.1	37.0
2	il12-sub5-exp2-fuzzy	sub5	Google search, Classifier	60.1	36.1
2	il12-sub5-exp3-exact	sub5	Google search, BERT, Rule	60.1	36.9

Table XIII
IL12 NER-ONLY SUBMISSIONS. A SMALL NUMBER OF THESE (MARKED WITH *, AND PRESENT IN TABLE XII) ALSO HAD EDL PREDICTIONS. ALL TRAIN IS: ALL IL12, TGL, SWA, ENG. RECALL THAT TWITTER RULES WERE USED AS POST-PROCESSING FOR ALL SUBMISSIONS (UNLESS OTHERWISE NOTED).

CP	Submission Name	Model	Data	Post-processing	NER	NER (IL)
2	il12-sub2-ner	fBERT	All Train	self	71.2	77.6
2	il12-sub3-exp1-exact*	cBERT	All Train	– (no twitter)	50.1	56.7
2	il12-sub3-exp2-exact*	cBERT	All Train	self	61.1	79.5
2	il12-sub3-exp3-exact*	cBERT	All Train	self	61.1	79.5
2	il12-sub4-ner	cBERT	Mixed IL data	self	**72.1**	79.5
2	il12-sub5-ner*	cBERT	Dense IL data	self	70.8	76.5
2	il12-sub6-ner	CCG+cBERT	All Train	self	71.8	79.0
2	il12-sub6-ner-fixed	CCG+cBERT	All Train	self	72.0	79.4
2	il12-sub7-ner	cBERT	All Train	no-bannawag	71.9	79.4
2	il12-sub8-ner	cBERT	All Train	entity list + self	71.9	79.2
2	il12-sub10-ner	cBERT	All Train	entity list	**72.1**	**79.7**
2	il12-sub11-ner	CCG	All Train	self	68.8	72.3
2	il12-sub18-ner	cBERT	Only IL data	self	70.6	76.2
2	il12-sub19-ner	fBERT	All Train (no dev1)	self	70.5	76.0
2	il12-sub20-ner	cBERT	All Train (no dev1)	self	71.7	78.8
2	il12-sub21-ner	cBERT	Mixed IL data (no dev1)	self	71.9	79.1
2	il12-sub22-ner	cBERT	Dense IL data (no dev1)	self	71.2	77.4

all sentences with 5% of tokens being entities and discarding all other sentences. This was our *Dense IL* submissions. The *Mixed IL* submission just added the *Dense IL* data to the original data. Examining submissions 3-5 in Table XI, we see that the *Mixed IL* significantly outperforms the original. Looking at the details of these scores, it turns out the mixed data produced significantly higher precision while harming the recall slightly. This contradicts our intuitions, and the reasons are unclear.

In both languages, we can see the comparative performance of including related languages in the training data by comparing sub3 (all training data) and sub18 (IL only data). For IL11, the improvement is very small (0.4F1), suggesting that IL11 is unique, and difficult to transfer to. Given prior work that suggests multi-source contextual training is valuable [14], and the large improvements we saw from related languages on our development sets (Tables V and VI), we found this to be a surprising result. For IL12, we see the opposite, showing nearly 3 points improvement from using only IL data (submission 18).

For IL12, one CP1 submission (*mbert_eng*) used only English, Spanish, Indonesian, and Tagalog data (and a very small amount of IL12 data) in training, along with the original mBERT model. This achieved the top score in the evaluation for checkpoint 1 (All, and IL only). Surprisingly, this also achieved an IL12 score of 70.9, only 9 points F1 lower than the final score. This is likely a combination of similarity between languages, ability of mBERT to generalize cross-lingually, and shared vocabulary and capitalization conventions.

As discussed in section IV-A2, we also made submissions using the non-neural Cogcomp NER system ensembled with the neural method (sub6-fixed in both languages), and on it's own (sub11 in both languages). When ensembled with the neural method, the performance is slightly below the neural score (sub3), reflecting how the ensembling strategy favors the neural method. On it's own, we see that the performance of the CCG model is substantially below the best neural counterpart (see the IL only scores), a gap of 12 points for IL11, and 6 points for IL12. This is consistent with prior intuitions suggesting that contextual embeddings for low-resource languages give improvements of about 10 points F1.

V. SITUATION FRAME

A. Native Informant Use

Entailment Approach: For our entailment approach, we built hypotheses for each of the 11 Situation Frame Types. For example, "(food): The people there are in need of food." In our first Native Informant (NI) slots, for each language, we then requested that the NI translate these hypotheses, listed in XVII, into the target languages (IL11 and IL12). The rest of this session was used to have them translate a set of approximately 130 disaster related words we compiled.

Then, we used the translated disaster related words above to rank the documents of set 1 by the quantity of disaster-related words in the text and presented them to the NIs in the remaining sessions. The NIs were asked to select the disaster type that each document was most related to, or select None if the document was out of domain. This data was used to further train some of our models prior to submission.

B. Annotated Datasets

- **BBN annotated data** is a multi-label SF typing dataset, with size of 4k examples. About half are "None" type.
- We also collected the gold annotated data from previous low-resource languages, including **IL3**, **IL5**, **IL9**, **IL10**, and **Mandarin**. For each IL, we use the released SF types and the English translated text as annotated instances.

The statistics of all annotated data are listed in Table XIV.

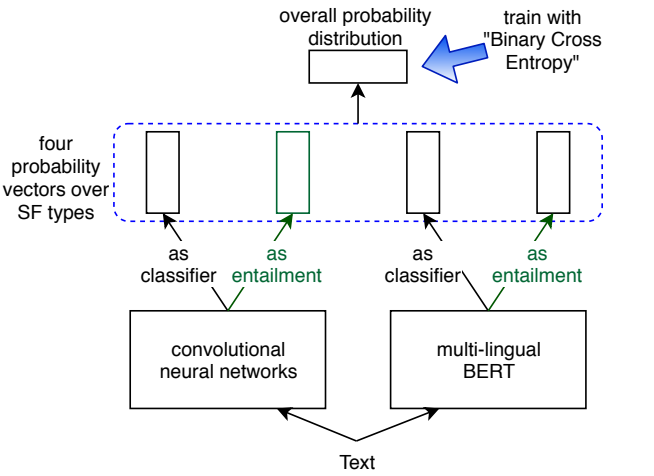
Table XIV
COLLECTED ANNOTATED ENGLISH TRAINING DATA FOR SF

	BBN	IL3	IL5	IL9	IL10	Mandarin
size	4,000	586	1,088	152	226	128

Table XV
IL11 SF SUBMISSION SYSTEM DESCRIPTION AND RESULTS (IL ONLY, CHECKPOINT 2)

	use BERT?	cBERT	bilingual_emb_version	#edl	test_input_type	F1 (IL type)	F1 (type+place)	nDCG (altref)	nDCG
#0	N	1M on IL11	dictionary with pair of single words	0	il11 text	34.09	14.87	3.24	1.36
#1	N	1M on IL11	dictionary with pair of single words	0	il11 text + MT	37.35	**16.82**	9.66	8.38
#2	N	1M on IL11	dictionary with pair of single words	0	MT	36.14	15.10	8.64	9.47
#3	N	1M on IL11	dictionary with pair of single words	1	il11 text	34.27	14.06	2.88	0.66
#4	N	1M on IL11	dictionary with pair of single words	1	il11 text+ MT	**37.80**	15.88	9.44	8.72
#5	N	1M on IL11	dictionary with pair of single words	1	MT	36.30	14.21	8.82	9.22
#6	N	1M on IL11	full dictionary	0	il11 text	33.16	13.75	16.79	20.19
#7	N	1M on IL11	full dictionary	0	il11 text + MT	35.97	15.62	8.49	1.12
#8	N	1M on IL11	full dictionary	0	MT	33.77	14.66	11.63	0.0
#9	N	1M on IL11	full dictionary	1	il11 text	33.38	12.94	**17.21**	**20.29**
#10	N	1M on IL11	full dictionary	1	il11 text + MT	36.32	14.57	4.66	1.12
#11	N	1M on IL11	full dictionary	1	MT	34.12	13.58	11.63	0.0
#12	Y	1M on IL11	dictionary with pair of single words	0	il11 text+ MT	31.20	10.83	11.63	0.0
#13	Y	1M on IL11	full dictionary	0	il11 text	32.24	11.12	11.63	0.0
#14	Y	1M on IL11	dictionary with pair of single words	0	il11 text+ MT	25.78	10.38	11.63	0.0
#15	Y	1M on IL11	full dictionary	0	il11 text+ MT	32.86	11.36	11.63	0.0
#16	Y	1M on IL11	dictionary with pair of single words	0	MT	31.35	12.11	11.63	0.0
#17	Y	1M on IL11	full dictionary	0	MT	32.04	11.87	11.63	0.0
			max released officially			37.80		37.09	38.16
			med released officially			17.95		13.25	15.61

Figure 2. SF System Architecture



C. System

Bilingual Embedding. To build bilingual word embeddings for English and the ILs, we first generate monolingual word embeddings for English and the ILs, respectively. For English, we always download the pre-trained 300d word2vec embeddings[12]. For each IL, we collect all the monolingual raw text from set 0, set 1, and set E, then train word2vec with hyperparameters: window 10, iteration 20.

Based on the resulting monolingual embeddings, we run BiCCA [3] along with the provided Eng-IL dictionary in set 0. BiCCA, in principle, can only use the pairs of single words in the dictionary.

Therefore, we build **two versions of the dictionaries**: one is to extract the pairs of single words from the original dictionary; the other is to convert the pairs with phrases into pairs of single words by picking up one word from the phrase iteratively – this means an original phrase pair with n:m will generate $n \times m$ pairs of single words. This will enlarge the resulting dictionary dramatically, and influence the final quality of the bilingual word embeddings.

Representation Learners

We treat the SF problem as classification as well as textual entailment. The benefit of exploring entailment is to make use of rich entailment datasets. We convert the SF types into hypotheses, as shown in Table XVII.

As Figure 2 shows, our SF system consists of four representation learners:

a) Convolutional neural networks as classifiers: The system takes the input text and outputs the probability distribution over the 12 SF types ("None" type included). Convolutional neural networks are well known for the keyword-based features [17], and they can also detect global features if attention mechanisms are incorporated [18].

[12]https://code.google.com/archive/p/word2vec/

Table XVI
IL12 SF SUBMISSION SYSTEM DESCRIPTION AND RESULTS (IL ONLY, CHECKPOINT 2)

	use BERT?	cBERT	bilingual_emb_version	#edl	test_input_type	F1 (IL type)	F1 (type+place)	nDCG (altref)	nDCG
#0	N	1M on IL12	dictionary with pair of single words	0	il12 text	20.79	5.34	2.99	2.35
#1	N	1M on IL12	dictionary with pair of single words	0	il12 text + MT	25.82	6.59	7.92	1.91
#2	N	1M on IL12	dictionary with pair of single words	0	MT	31.57	8.55	1.84	4.19
#3	N	1M on IL12	full dictionary	0	il12 text	27.24	6.78	6.91	4.33
#4	N	1M on IL12	full dictionary	0	il12 text + MT	30.82	7.73	5.90	7.28
#5	N	1M on IL12	full dictionary	0	MT	20.89	6.57	3.62	3.18
#6	Y	1M on IL12	dictionary with pair of single words	0	il12 text	15.35	4.20	5.49	5.16
#7	Y	1M on IL12	dictionary with pair of single words	0	il12 text + MT	22.94	5.12	5.67	8.21
#8	Y	1M on IL12	dictionary with pair of single words	0	MT	24.23	5.84	22.18	21.23
#9	Y	1M on IL12	full dictionary	0	il12 text	0.0	0.0	5.99	5.74
#10	Y	1M on IL12	full dictionary	0	il12 text + MT	23.83	5.26	**23.88**	21.23
#11	Y	1M on {Eng, IL12 and simi-lang}	dictionary with pair of single words	0	il12 text	23.43	5.66	22.18	21.23
#12	Y	1M on IL12	full dictionary	0	MT	25.57	6.45	22.18	21.23
#13	Y	1M on {Eng, IL12 and simi-lang}	dictionary with pair of single words	0	il12 text+ MT	22.85	5.20	22.18	**25.65**
#14	Y	1M on {Eng, IL12 and simi-lang}	full dictionary	0	il12 text	18.81	5.58	22.18	21.23
#15	Y	1M on {Eng, IL12 and simi-lang}	dictionary with pair of single words	0	MT	24.53	5.90	22.18	21.23
#16	Y	1M on {Eng, IL12 and simi-lang}	full dictionary	0	il12 text + MT	24.87	6.08	22.18	21.23
#17	Y	1M on {Eng, IL12 and simi-lang}	full dictionary	0	MT	30.07	7.66	22.18	24.63
#18	N	1M on IL12	dictionary with pair of single words	1	il12 text	22.92	6.13	8.40	0.0
#19	N	1M on IL12	dictionary with pair of single words	1	il12 text + MT	29.31	8.14	22.18	0.0
#20	N	1M on IL12	dictionary with pair of single words	1	MT	33.53	**9.73**	5.18	1.34
#21	N	1M on IL12	full dictionary	1	il12 text	31.42	8.76	8.08	5.88
#22	N	1M on IL12	full dictionary	1	il12 text + MT	33.57	9.50	8.83	3.40
#23	N	1M on IL12	full dictionary	1	MT	**34.13**	9.25	22.18	21.23
#24	Y	1M on IL12	dictionary with pair of single words	1	il12 text	20.71	4.59	9.56	9.16
#25	Y	1M on IL12	full dictionary	1	il12 text	21.16	4.75	10.39	12.79
#26	Y	1M on {Eng, IL12 and simi-lang}	dictionary with pair of single words	1	il12 text	0.0	0.0	12.78	13.44
#27	Y	1M on {Eng, IL12 and simi-lang}	dictionary with pair of single words	1	il12 text + MT	24.80	5.98	7.84	12.47
#28	Y	1M on {Eng, IL12 and simi-lang}	dictionary with pair of single words	1	MT	27.04	6.86	11.03	9.82
#29	Y	1M on IL12	full dictionary	1	il12 text + MT	24.29	5.59	16.20	12.54
#30	Y	1M on IL12	dictionary with pair of single words	1	il12 text + MT	22.93	5.17	15.35	12.56
#31	Y	1M on IL12	full dictionary	1	MT	26.83	6.78	16.17	14.31
#32	Y	1M on IL12	dictionary with pair of single words	1	MT	25.58	6.39	14.22	13.81
			max released officially			38.97		39.05	25.65
			med released officially			22.77		9.02	9.82

Table XVII
CONVERTING SF TYPES TO HYPOTHESES

	type	premise example
need	med	people need medical assistance
	search	some people are missing or buried, we need to provide search and rescue
	food	people are in shortage of food
	infra	the infrastructures are destroyed, we need to build new
	water	people are in the shortage of water
	shelter	Many houses collapsed and people are in desperate need of new places to live
	utils	The water supply and power supply system is broken, and basic living supply is urgently needed.
	evac	This place is very dangerous, and it is urgent to evacuate people to safety.
issue	terrorism	There was a terrorist activity in that place, such as an explosion, shooting
	regime change	Regime change happened in this country
	crime violence	There was violent criminal activity in that place

Table XVIII
SF SYSTEM PERFORMANCE ON IL9 AND IL10

system	IL9		IL10	
	F1	weighted F1	F1	weighted F1
top record	31.03	–	24.09	–
Attentive Convolution	32.30	46.71	28.60	45.44
cBERT				
1M on IL	29.44	44.60	14.11	29.86
1M on {IL, En, sim-L}	24.32	38.14	16.95	33.18
Attentive Conv. & cBERT				
1M on IL	39.10	53.34	25.79	42.17
1M on {IL, En, sim-L}	35.94	50.43	27.01	43.41
+SF gold data from {il9, il10, il5, il3, Mandarin}	48.32	60.78	40.44	55.51

b) Convolutional neural networks for entailment: Convolutional neural networks can also work for entailment if phrase-level reasoning plays a role [19]. We have two copies of the same convolutional component, one working on the premise, the other working on the hypothesis. It is trained for "yes or no" for each premise-hypothesis pair. We use the probabilities of entailment as the output probability vector.

c) cBERT as classifier: We treat (multi-lingual) BERT as a new classifier – it takes text as input, then outputs a representation. On the top of cBERT, we stack a classifier.

d) cBERT for entailment: Similar to attentive convolution [18], BERT originally works on local regions (i.e., single tokens), but the attention mechanisms enable it to encode global information. BERT can work on single sentences as well as a pair of sentences. For entailment, we apply cBERT on the premise-hypothesis pairs and get the resulting representation. The remaining parts are the same as the above "convolution for entailment".

In total, our SF approach includes four subsystems, each providing one probability vector. We do an element-wise average to get the overall probability vector. The whole system is optimized by minimizing the binary cross-entropy.

D. Submissions and Results on IL11 & IL12

Our submissions take the outputs of EDL and MT as inputs. EDL provides the information of detected entities, and MT provides the English translation of the original IL text. TIn total, we tried two EDL outputs and three ways of incorporating MT and IL text: IL text only, concatenating IL text with MT, MT only.

Table XV lists all 18 submitted systems with their respective setups for the IL11 language. Table XVI lists all 33 submitted systems with their respective setups for the IL12 language.

Only checkpoint 2 results of on low-resource text (IL) are reported since this is our target.

In both IL11 and IL12, the best typing performance always comes from "full dictionary" in which we split phrase pairs into single-word pairs. These results show its effectiveness in learning higher-quality bilingual word embeddings.

In most cases of IL11, combining IL and MT as input improves the tying performance. However, in IL12, using only MT seems better than IL and their combination.

E. Experiments on IL9 and IL10

Due to lack of time for testing all of our setups in IL11 and IL12, especially those using BERT, we also test our setups on IL9 and IL10. In Table XVIII, we notice that attentive convolution works consistently well in both IL9 and IL10, but the cBERT performed poorly in IL10. The results show that the combination of attentive convolution and cBERT may be less promising than just attentive convolution. However, the same combination promotes the performance in IL9 dramatically, especially when cBERT is trained on the IL with 1M iterations. In both languages, we achieved big improvements by collecting the SF gold data on translated IL text from prior ILs including Mandarin, IL3, IL5, IL9, and IL10.

VI. CONCLUSION

We have described details for high-performing systems for named entity recognition, entity linking, and situation frame tasks developed as part of the LoReHLT 2019 evaluation. One innovation over prior years is that each of these tasks took advantage of cross-lingual contextual embeddings for strong performance. We hope that the innovations and procedures outlined in this report can be used by others to help the rapid development of low-resource NLP systems in surprise languages.

VII. ACKNOWLEDGEMENTS

Many thanks to Jordan Kodner at the University of Pennsylvania for help with various tasks including language facts, morphology, and rescripting. Also many thanks to Chris Callison-Burch and several of his summer interns for their help in annotation. This work was supported by Contracts HR0011-15-C-0113 and HR0011-18-2-0052 with the US Defense Advanced Research Projects Agency (DARPA) and by a Focused Award from Google. This project is also supported with Cloud TPUs from Google's TensorFlow Research Cloud (TFRC).

REFERENCES

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*, pages 462–471, 2014.

[4] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

[5] Ulf Hermjakob, Jonathan May, and Kevin Knight. Out-of-the-box universal romanization tool uroman. In *Proceedings of ACL 2018, System Demonstrations*, pages 13–18, 2018.

[6] Daniel Khashabi, Mark Sammons, Ben Zhou, Tom Redman, Christos Christodoulopoulos, Vivek Srikumar, Nicholas Rizzolo, Lev Ratinov, Guanheng Luo, Quang Do, Chen-Tse Tsai, Subhro Roy, Stephen Mayhew, Zhili Feng, John Wieting, Xiaodong Yu, Yangqiu Song, Shashank Gupta, Shyam Upadhyay, Naveen Arivazhagan, Qiang Ning, Shaoshi Ling, and Dan Roth. Cogcompnlp: Your swiss army knife for nlp. In *11th Language Resources and Evaluation Conference*, 2018.

[7] Xuezhe Ma and Eduard H. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

[8] Stephen Mayhew. TALEN: Tool for Annotation of Low-resource ENtities. In *ACL Demonstrations*, 2018.

[9] Stephen Mayhew, Chase Duncan, Mark Sammons, Chen-Tse Tsai, Xin Li, Haojie Pan, Sheng Zhou, Jennifer Zou, and Yangqiu Song. University of Illinois LoReHLT17 Submission.

[10] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL-09*, pages 147–155, 2009.

[11] Stephanie Strassel and Jennifer Tracey. Lorelei language packs: Data, tools, and resources for technology development in low resource languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3273–3280, 2016.

[12] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 142–147. Edmonton, Canada, 2003.

[13] Chen-Tse Tsai, Stephen Mayhew, Yangqiu Song, Mark Sammons, and Dan Roth. Illinois CCG LoReHLT 2016 Named Entity Recognition and Situation Frame Systems. *Machine Translation*, 2018.

[14] Tatiana Tsygankova, Stephen Mayhew, and Dan Roth. BSNLP2019 shared task submission: Multisource neural NER transfer. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 75–82, 2019.

[15] Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*, 2011.

[16] Shijie Wu and Mark Dredze. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. *ArXiv*, abs/1904.09077, 2019.

[17] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of CNN and RNN for natural language processing. *CoRR*, abs/1702.01923, 2017.

[18] Wenpeng Yin and Hinrich Schütze. Attentive convolution: Equipping cnns with rnn-style attention mechanisms. *TACL*, 6:687–702, 2018.

[19] Wenpeng Yin, Hinrich Schütze, and Dan Roth. End-task oriented textual entailment via deep explorations of inter-sentence interactions. In *Proceedings of ACL*, pages 540–545, 2018.