India-centric Never Ending Language Learning (INELL): Milestone 2 Report

Machine And Language Learning (MALL) Lab
Contact: Dr. Partha Talukdar (ppt@iisc.ac.in)

Period: September 2017 - March 2018

In this report, we report developments on the following fronts during the second phase of the INELL project.

1. Scalable D-CAP Deployment
2. Increase in corpus size
3. Entity linker
4. Summary statistics of the collected corpus
5. Implementation of initial KG algorithms

Scalable D-CAP Deployment

We first begin with a brief overview of the Domain-specific Content Acquisition Platform (D-CAP), a critical component of the overall INELL system. D-CAP aims at building a domain-specific corpus out of the general Web. D-CAP is real-time, streaming, modular, and distributed.

A majority of state-of-the-art work on domain-specific crawling is proprietary in nature [1]. Published literature [7] in this area have focused on individually tuning various crawling aspects, e.g., concurrency, recall, precision, modularity, scheduling, and scalability. In contrast, the D-CAP platform attempts to jointly optimize all the previously mentioned aspects, while at the same time needing significantly lesser computational power. Apart from the architecture, D-CAP's novelty also lies in the use of a Knowledge Graph (KG) [3] for faster inferencing. This helps D-CAP in identifying relevant pages which are multiple hops away from the seed set, while carefully filtering out millions of directly-linked irrelevant pages.

D-CAP also employs Graph Databases and uses modified GraphQL [4] for faster traversals. We report on domain-specific crawling experiments using seed documents with different level of topic specificity. We find that while D-CAP remains within the topic radius, a general crawler tends to drift away quickly. Our results demonstrate that D-CAP is very effective in creating large domain-specific corpora while using only modest computing. We also report on the quality of the collected corpus and provide some examples of the kind of structured information we can extract from the clean corpus.

Architecture and Deployment

We chose **Docker containers** for better scalability and efficiency in D-CAP. Given the modular design, it was easy to make Docker containers for each module separately so that the a Single Container Single Responsibility condition is met. The following docker containers were prepared:

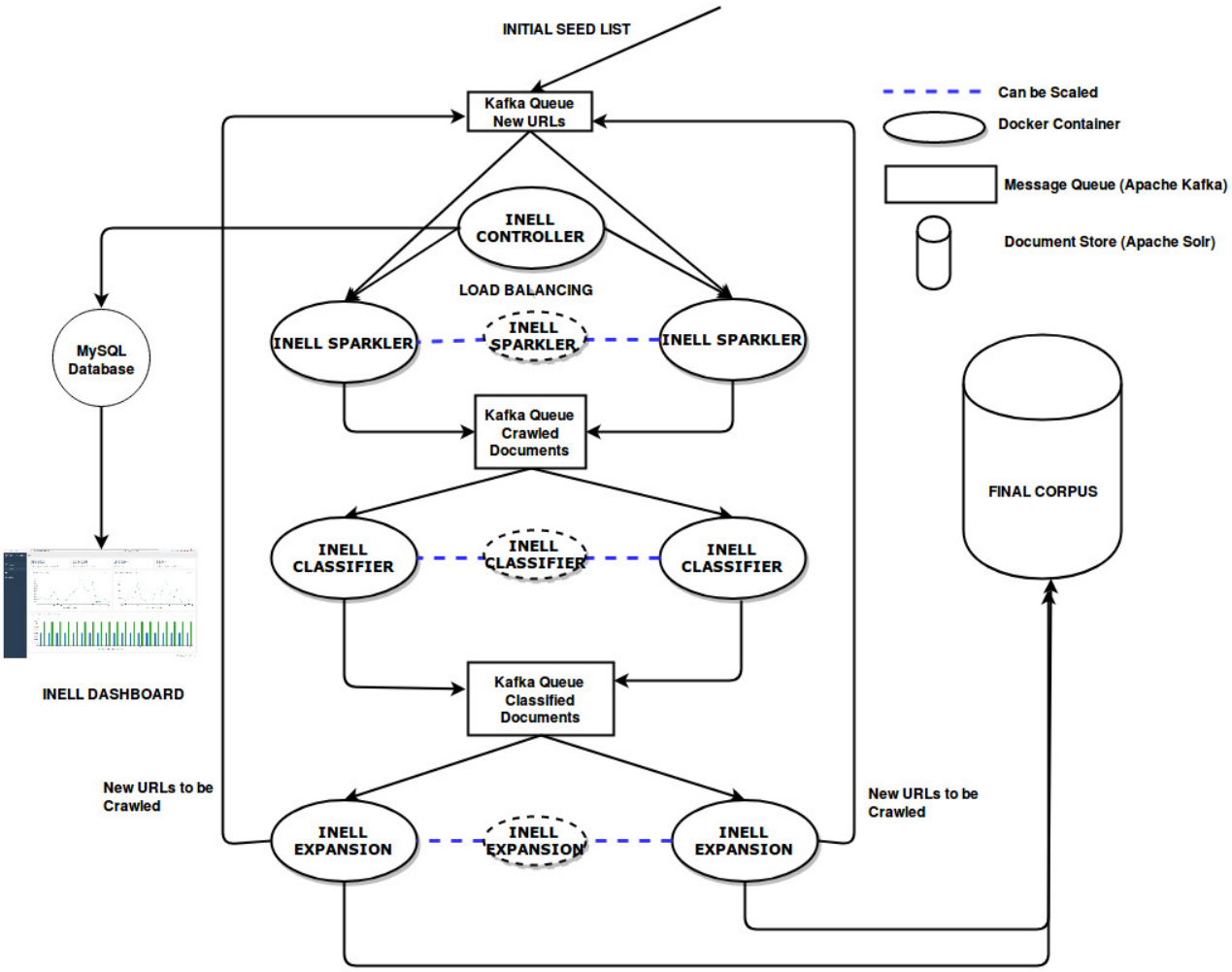

Figure 1: Scalable D-CAP architecture

- **INELL CONTROLLER:** The main master container that hosts the dashboard, acts as beacon for other services (sparkler, classifier, expansion) and handles load balancing for sparkler modules.
- **INELL SPARKLER:** The sparkler container picks urls from the kafka queue, crawls those urls and pushes them into another kafka queue.
- **INELL CLASSIFIER:** The classifier container reads data from the kafka queue where sparkler pushes the content of the webpages that have been crawled and uses the wikipedia category graph for classification. The classifier classifies the document as Indian or Non-Indian and accordingly pushes the document into the solr collection.
- **INELL RNN CHARACTER CLASSIFIER:** We have also included a character level classifier that predicts the probability of a given entity string to be Indian. We provide the trained model file as well as the executable to run the rnn character level classifier.

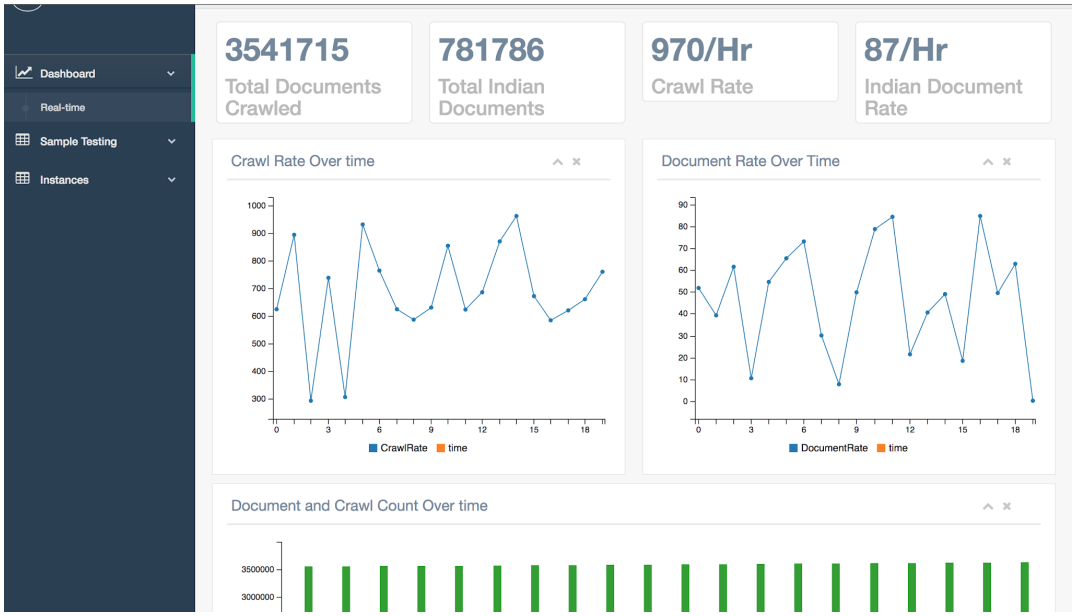

Figure 2: Snapshot of the D-CAP dashboard

- **INELL EXPANDER:** The expander module makes a collection of all the outgoing links from the selected Indian documents. It then injects them into sparkler for crawling and the loop is thus completed, and the system reiterates.

All the above specified docker containers have been deployed via **Kubernetes** which gives the following advantages

- **Resiliency:** Automatic container restart on failure.
- **Easy scaling:** The number of containers can be scaled up / down with one click.
- **Global Configuration:** No need to configure each module separately. The same configuration is reflected in all containers (even if configuration is changed dynamically).

Additional Features

1. **Dashboard for easy monitoring:** We have added a dashboard module to D-CAP to make monitoring of crawl rate / total documents / total Indian documents easy. The dashboard also has functionality to search for documents crawled between timestamps.
2. **Status Alerts:** We have added a feature using which when a component is unresponsive or is killed off, an alert email is sent to the user for information. Automatic restarts are taken care of and in the event that one of the INELL sparkler module is killed, the respective hostnames allocated to the killed sparkler module are reallocated to other sparkler containers.
3. **Documentation:** We have added step-by-step documentation on building the project, creation of docker images, kubernetes deployments, setting up of the Solr / Kafka multinode clusters, links to tutorials / blogs to get a broad level idea of the technologies used.

4. **Upgraded Sparkler**: We have upgraded Sparkler to newer version resulting in increased stability.

5. **Bots in Sparkler:** We have added bots to the Sparkler module to make sure that websites don't block the crawler. The has resulted is less blockage and higher crawl rate.

Corpus Size

The D-CAP is a composite system which processes data in an assembly line manner. To enable D-CAP act as a large-scale harvester, we have kept the vertical compute requirement to the modest, while at the same time allowing an option to add multiple such instances in parallel, thereby scaling up the system compute horizontally.

At present, the Indian corpus contains **2.5M documents**, out of a total **6.5M crawled documents (25% selectivity)**. The current corpus is a mixture of the web crawl as well as documents obtained from the web crawl repository Common Crawl which is updated on a monthly basis. In order to fetch more in-domain documents as the iterations increases, we search for hostnames of the in domain documents in the Common Crawl repository and pass them on to the classifier. Of the 2.5M documents, about 2M of them were fetched from Common Crawl and 500K by the crawler pipeline. The systems runs 24x7, the functioning of which is explained in the next section.

Entity Linker

Entity linking is the process of identifying and linking entities mentioned in a document to the entities of an existing knowledge base. It is used for disambiguating entities and has applications in information retrieval, search and related areas. For the existing setup, we use Wikidata as the knowledge base for linking entities mentioned in the documents.

Datasets

Although there have been entity linkers for Wikipedia and other KBs before, not much work has been done on linking entities to Wikidata [6]. Wikidata is of special interest to us because it has about 100 times more entities than Wikipedia and is entirely human curated. At present, there are no Wikidata annotated corpus available in the community, therefore, for evaluation we utilize Wikipedia and Freebase annotated corpus.

Methodology

Currently, our method gives an accuracy of 70% on Wikipedia corpus. To accomplish this, we use Stanford CoreNLPs NER classifier to extract the mentions of the entities and use the CrossWiki dataset to link the entity to the corresponding Wikidata entity which are already mapped to the Wikipedia entities.

Future Work

We plan to extend the current work across all the entities of Wikidata and also the crosswiki dataset with entities/mentions in Wikidata using various NLP techniques and deep learning methods.

Summary Statistics

This section describes the "all-pairs" data that CPL (Couples Pattern Learner in NELL) takes as one of its inputs, consisting of corpus statistics about the occurrence and cooccurrence frequencies of NPs (or NP pairs) and nearby textual context patterns.

A series of MapReduce jobs [2] are used to distill a representative set of vectors and matrices out of the large corpus mentioned in previous section. The vectors and matrices are meant to be small enough to be manipulated on a single server.

Steps of Operation:

1. Turn the corpus into a set of unique POS-tagged sentences
2. Generate matrices from sentences by running following sequence of jobs
    - **AllPairsExtractor:** Identify NPs and adjacent context patterns in each sentence, and output these pairs. This is done by an extensive system of rules and filters based primarily on POS tags.
    - **ReduceJob:** Aggregate the output of AllPairsExtractor and output a set of unique pairs along with a count of how many times each was found.
    - **AllPairsContextsFilter:** Apply thresholds to filter the output from ReduceJob based on occurrence and cooccurrence counts.
    - **AllPairsCountVectorsGenerator:** Extract from the above output a vector of (NP, count) pairs, a vector of (context, count) pairs, a matrix of NP by context where each cell is a cooccurrence count, or a matrix of context by NP where each cell is a cooccurrence count. This is run 4 times in order to obtain all 4.
    - Rerun the above using NP pairs instead of NPs so as to generate output for relations instead of categories.

      Apart from the summary statistics, we also calculate entity frequencies on a subset of the dataset processed via CoreNLP. Preprocessing is done on a subset of 100k indian documents and frequencies of all linked entities have been calculated. The resulting entities are cleaned to remove numeric entities and the resulting file is provided in the DVD.

Current size of this all-pairs statistics is 352MB generated from 2 million text documents. A manual inspection of the all-pairs data shows there is good representation of entities of our interest in the corpus. Statistics of the all-pairs data are presented below:

1. Number of Category Noun Phrases - 126960
2. Number of Category NP pairs - 124302
3. Number of Relation Noun Phrases - 1280798
4. Number of Relation NP - 968702

KG Extraction Algorithms

Distant Supervision

Knowledge Graph construction is an important task in NLP as knowledge graphs have a wide range of applications in problems such as question answering, text classification etc. We primarily focused on using Distant Supervision (DS) for construction of an Indian Knowledge graph since it requires low amounts of training data. In this technique, we use some existing knowledge base with which we align the corpus to obtain large amounts of training data for our algorithms.

Specifically, we extract entity pairs from the knowledge graph that are of importance and find the relations that exist between them in the knowledge graph. Let e1 and e2 be the two entities that we are interested in and r be the relation between them according the existing knowledge base. We then label all the sentences that contain e1 and e2 to be instances of the relation r.

We then train a classifier to predict the type of relation expressed by the sentence between the two entities. The above approach of labeling all sentences that contain e1 and e2 as instances of the relation r is known as the distant supervision assumption originally proposed by Mintz et al., 2009 [8]. Since this is a very strong assumption, it is often weakened by considering a Multi Instance Multi Label (MIML) setting in which we construct bags of sentences. Each bag contains the sentences that contain the same entity pair. Our classifier is then trained to make predictions on bags rather than sentences. All of the current state-of-the-art algorithms use the bags approach for DS.

Seeds

In order to use DS for KG construction, we need an existing knowledge base to align the corpus with. Since such a Knowledge Base does not exist, we manually select some seeds for preliminary KG construction. These seeds are selected in a data driven manner, i.e., we select those entities of interest which are supported by the extracted corpus. These seeds are then used to create the training data using the distant supervision assumption. Once we have formed the bags, we apply state-of-the-art distant supervision algorithms like PCNN, BGWA, EA etc, to obtain the extractions.

Methods: PCNN, BGWA, EA, and Ensemble

Piecewise Convolution Neural Network (PCNN) [9] is a recently proposed distant supervision model. The PCNN model summarized in Figure 3. However, not all words in the sentence contributes equally in deciding the relation expressed in the sentence. To leverage this property, we recently proposed a word attention based model [5] in which the model attends to each word of the sentence and uses that to decide the relation expressed in the sentence. We use a state of the art GRU to capture long-term dependencies. Figure 4 summarizes the model. We hope to utilize these models as part of the INELL project.

Results

In the preliminary stage, we have experimented with a few pairs of entities. We experimented with the relation **terrorist_organization/country/operates_in**. As seeds for this relation, we picked **ISIS operates_in Iraq** and **ISIS operates_in Syria**. To test whether the model learns the properties of the relation, we test it on the entities **Al-Qaeda**, **Afghanistan** which we know should show the selected relation. We found that our model successfully predicts the operates_in relation between Al-Qaeda and Afghanistan.

Table 1: Sample Top Extractions Per Relation

operates_in	leader_of	works_for
Jamaat-e-Islami operates_in Israel	Modi leader_of India	al-Zawahiri works_for al-Qaeda
Jaish-e-Mohammad operates_in Jammu	Trump leader of US	Osama works_for Taliban
ISIS operates_in Sudan	Assad leader_of Syria	Hafiz_Saeed works_for Lashkar-e-Taiba
taliban operates_in Afghanistan	Erdogan leader_of Turkey	Zakir_Musa,works_for,al-Qaeda

Algorithm: PCNN

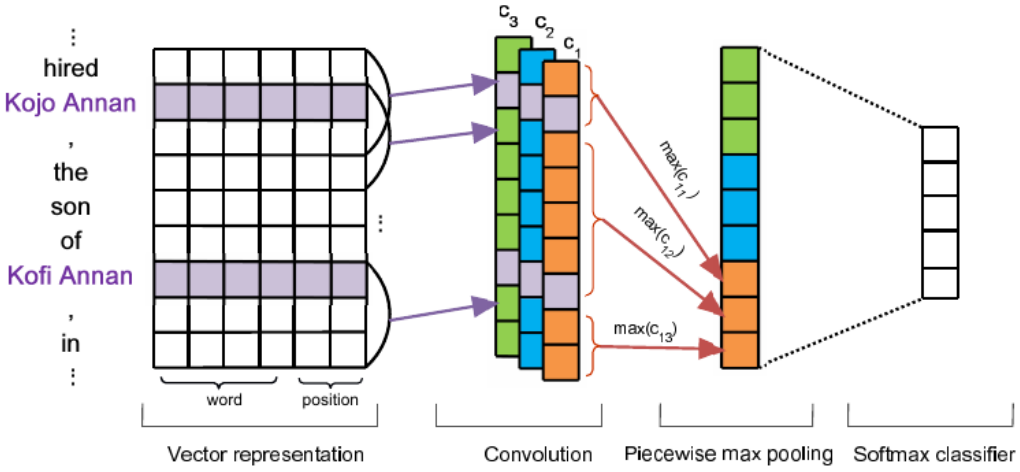

Figure 3: Piecewise Convolution Neural Network (PCNN) model

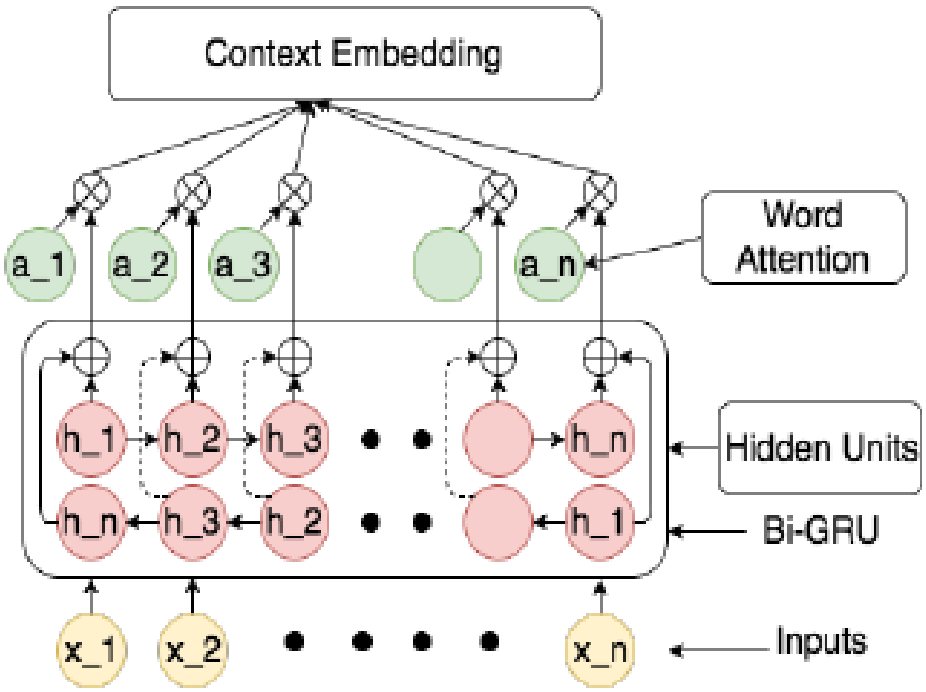

Figure 4: Our proposed BiGRU-based word attention model for Distant Supervision [5].

Seeds: Around 200 per relation.

Seeds creation process: We went through Wikipedia to find out terror organizations, their known personnel, list of countries and their known leaders. We then took a cross product according to relation type, e.g., for the *works-for* relation, we took the cross product of all terrorists and known organizations. If any sentence in the corpus mentioned both of them, then it was marked as a positive sentence for the relation.

Example Sentences

– The terrorists, who attacked the military base in Jammu and Kashmirs Uri, belonged to Pakistans banned terror outfit Jaish-e-Mohammed (JeM).
– The so-called black day coincides with Lashkar-e-Taiba founder Hafiz_Saeed, accused of masterminding 2008 Mumbai terror strikes, leading a Kashmir Carvan into Islamabad from Lahore to protest against the unrest in the Indian state.
– Several bombings have targeted Syrian cities held by the government of President Bashar al-Assad in recent months.

Future work

In the next stage of the project, we plan to extend our work for many more entity pairs and extract meaningful relationships between important entities. This will hopefully help us to construct a useful knowledge graph.

References

[1] S. Chakrabarti, B.E. Dom, and M.H. van den Berg. System and method for focussed web crawling, July 9 2002. US Patent 6,418,433.

[2] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[3] Google. Freebase data dumps. `https://developers.google.com/freebase/data`.

[4] Manish Jain. Dgraph. `https://github.com/dgraph-io/dgraph`, 2016.

[5] Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. Improving distantly supervised relation extraction using word and entity based attention. 2017.

[6] Xiao Ling, Sameer Singh, and Daniel S Weld. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3:315–328, 2015.

[7] Luis A Lopez, Ruth Duerr, and Siri Jodha Singh Khalsa. Optimizing apache nutch for domain specific crawling at large scale, 2015.

[8] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[9] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, 2015.