CSE 461 Software Engineering

Spring 2020

NLP toolkit as a service

Project High Level Design Documentation

Saumitra Yadav (201507647)
Sunil Gundapu (2018701022)
Prashant Kodali (2018801011)
Neha Motlani (20161004)
Ananya Mukherjee (2018801009)
Hiranmai Sri Adibhatla(2018900044)

Table of Contents

I. Introduction

In the course of any NLP application development, there are many basic steps which are common across all NLP tools. These may be as basic as Tokenization, Named Entity Recognition (NER), Text Representations (Embeddings) and extend to as complex tools as Sentiment Analysis, Machine Translation etc. NLP applications typically are visualized as a pipeline, where the initial few blocks are preprocessing steps, like tokenization, NER, parsing etc, followed by complex blocks of learnt representations (word2vec, BERT etc), models (Neural, statistical, ML algorithms).

In this pipeline we see a lot of reuse. The basic building blocks are required in every application in some form or the other. And these blocks are crucial as well. In the world of NLP it is a common occurance where a simple tweak in tokenization affects your results significantly.

In such a scenario, if there are multiple teams working within an organization (industry or research) it is prudent to share the work that is already done. It avoids rework, exposes the developer to various ideas that may exist within the organization and potentially has the benefit of quick turnaround time if you are developing an application / prototype. For example: if a team wants to use word representation, like BERT, it is quite wasteful for every team to understand how BERT works, how to deploy and get its representations. Or word2vec, where every team is downloading the embeddings and using. This has a direct impact on computation as well as time resources.

This is where our motivation lies. To build an online portal, which is modular enough, so that newer applications can be published for everybody who needs that

knowledge / resources for their use. To build a system where these individual applications can be reused by someone else in a quick and efficient manner. This solution is aimed to address the transient nature of NLP as a field, where the research is moving fast, and applications development needs to try out multiple options to get an optimum solution.

II. Terminology

1. Service: Individual applications running in different machines/containers, offering specific use case: Tokenization, NER, Sentiment Analysis etc
2. Backend: Collection of all the published services.
3. Publisher: Developers who publish their services in the portal.
4. Users: Developers who want to look at the existing published services, for using these services for their applications.
5. Front End: The website used as a common place for aggregating, viewing all this information.

III. System Overview

Our proposed solution has following broad components:

1. A front end application
2. Backend individually deployed applications.

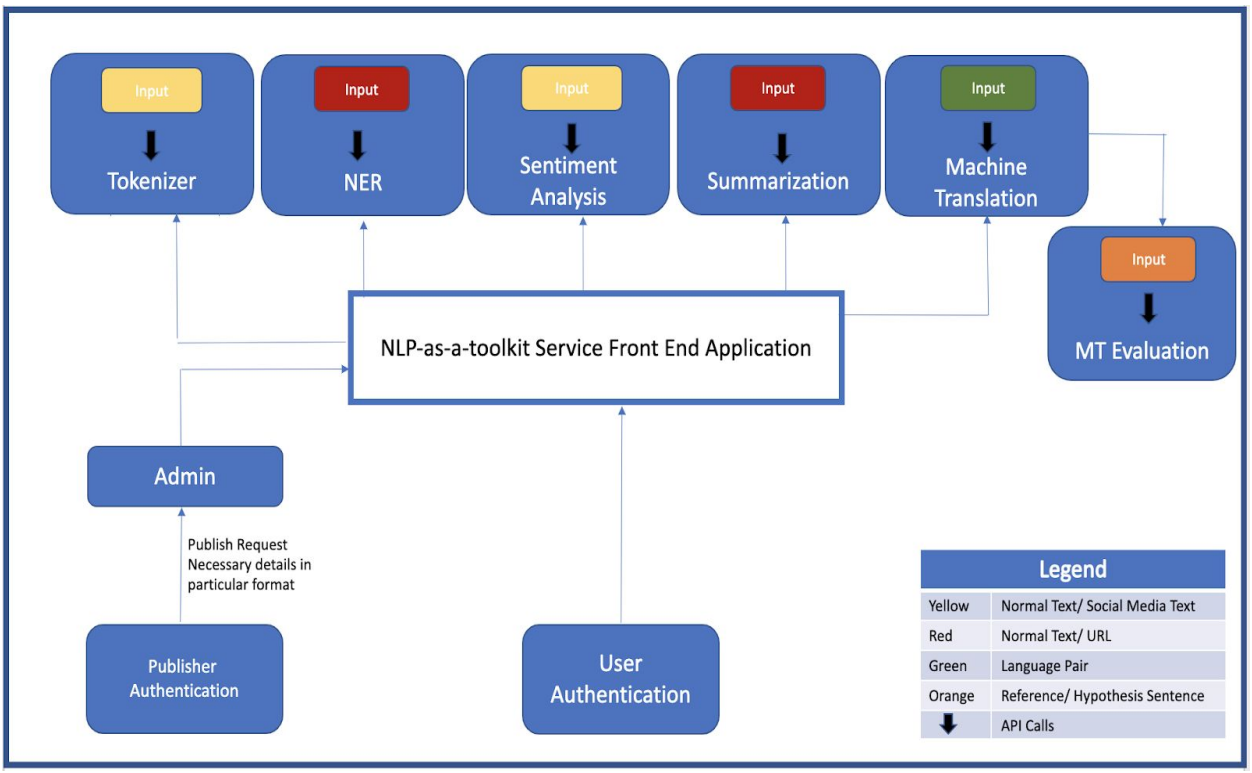
- **Front end applications** will aggregate all the information about the individual applications deployed, their specifications and a portal to try these applications - try out a sample or two before you use it. The portal is just for aggregating various services, and function as a library where developers can look at various implementations / options he has in front of him and choose what works best for him thus allowing the developer to build on previous efforts.

- **Individual applications** are deployed in different machines as services.These services expose their own APIs. These APIs are used by front end applications for demo purposes. Or these APIs can be called individual applications if a user intends to use them in his/her application.

  The individual applications are perceived as different machines and deployments for following reasons:

1. Extensibility : New application can be continuously published and integrated into the portal
2. Scaling: a particular service, if it is seeing more demand, should be scaled individually, leaving minimal to no impact on other services.
3. Managing Complexity: Individual blocks in a NLP pipeline can be complex. When the APIs are used it abstracts away the complexity from the end user, who in this case is a developer.
4. Reuse: Using basic building blocks like tokenizers, representations.
5. Maintainability: Any improvement or scaling will have impact on the individual blocks.
6. Modular: Individual services can be incrementally improved. Adding new features and improvements will not have any implication on the working of the service.

IV. System Architecture



1. **Front End application:**
   Major components of this application are:

1. User authentication
2. Publisher module: For publishing individual modules on the portail
3. Portal Design:
    a. Post login the user will see the available applications/ modules.
    b. Each module will lead to its own page, which will include:
        i. Broad information of the service: if a prediction model then the dataset information and accuracy measures, limitations of the solution.
        ii. Option to try it out
        iii. API and other details, if a developer wants to use it.

2. Individual Services:

Services offer specific APIs for that application. A service can offer multiple APIs. For example: Sentiment analysis would need different APIs for normal text, Social media text, language choice etc.

For Minimum viable products we are planning the following services.:

1. Tokenizer
2. NER
3. Transliteration
4. Generating BERT representations
5. Sentiment Analysis
6. Summarization
7. Machine Translation & Evaluation

These are the services that we are publishing in the portal. Other services can be published to the portal, because of its modularity. Details of aforementioned services are elaborated in Appendix A.

3. Technical and Deployment Choices

1. Individual services can be deployed in his/her choice of tech stack by the publisher. Ony condition is that the service should conform to API specs.
2. Applications can be run on local machines over LAN or cloud. Assessment of possible solutions in progress.
3. The Formulating Process for publishing and authentication is in progress.

4. Cons:

1. Single Point Of Failure at Front end level: Would need a distributed design to address this issue.
2. The MT system needs to retrain a system if we acquire new linguistic features.

V. Challenges Faced Till Now

Currently the team is discussing our options for deployment. Initially the plan was to deploy this over machines connected over LAN, which would serve the purpose of Minimum viable product demo as well. However, no that is under revaluation since the team is working from remote.

- In tokenization as service
  Handling Homonyms
  Rare Words Occurrence

- In NER as service
  - Foreign Words (Words which are not used very frequently these days, or words that are not heard by a lot of people, is another major challenge.)
  - Ambiguity and Abbreviations
  - Demands a high quality POS Tagger.

- In Text Representation as service
  - Word embeddings are dense vector representations of words in lower dimensional space. Word sense disambiguation is the most important challenge while creating representations for text / word.
  - BERT embeddings use a bi-directional transformers model to train on huge amounts of data, and it is difficult to use such large models under low latency constraints. May require may need (costly) GPU servers to serve at scale
  - Running models on devices like your smartphone also requires light-weight, responsive and energy-efficient models.

- In Sentiment Analysis as service
  - In Social media data we will see so many informal words like n8 (night), twrw (tomorrow)...etc. To handle this kind of words manually created a dictionary (informal/short words to formal words) of 200 words.

    - To handle the unseen words used the character level word embeddings.
    - Large amount of data available for Hindi-English Code-Mixed data.
- Summarization:
    - Will be deployed only for english. And will use sentence representation generation and KNN algorithm for generating sentences that best summarize the text.
    - Due to the limitations the team is facing, the initial plan of pointer-generator approach for summarization won't be implemented.
- In MT as service
    - Availability of good quality data for training MT systems for Indian Languages
    - Good linguistic tool for many indian languages

VI. Next Milestones

1. Social Media text tokenization and Sentence segmentation.
2. To reduce the size of these high computing models. To focus on a compression technique in which a small model is trained to reproduce the behavior of a larger model.
3. Handle the sarcasm and improve the f1-score of Code-Mixed Sentiment Analysis model.
4. Basic MT systems (based on two neural network architectures) for a language Pair.
5. Integrating UI, with all the individual services.

VII. References

01. Tokenization
    a. Jurafsky, Dan. *Speech & language processing*. Pearson Education India, 2000.
02. NER
    a. Ratinov, Lev, and Dan Roth. "Design challenges and misconceptions in named entity recognition." *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*. 2009.
    b. Hardeniya, Nitin. *NLTK essentials*. Packt Publishing Ltd, 2015.
    c. Schmitt, Xavier, et al. "A Replicable Comparison Study of NER Software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate." *2019 Sixth International*

Conference on Social Networks Analysis, Management and Security (SNAMS). IEEE, 2019.

03. Text Representation
    a. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.Paper presented at the NAACL-HLT
    b. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., . . . Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. CoRR, abs/1907.11692

04. BPE processing
    a. Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units." *arXiv preprint arXiv:1508.07909* (2015).
05. Sentiment Analysis
    a. Hutto, C. J. & Gilbert, E. (2014), VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text., *in* Eytan Adar; Paul Resnick; Munmun De Choudhury; Bernie Hogan & Alice H. Oh, ed., 'ICWSM' , The AAAI Press, .
    b. Yash Kumar, Vaibhav, Mrinal, Manish, and Philipp. “De-Mixing Sentiment from Code-Mixed Text.” *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop.*
    c. Pruthwik, Prathyusha, Pranav. “Code-Mixed Sentiment Analysis Using Machine Learning andNeural Network Approaches.”
06. Machine Translation & Evaluation
    a. Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473* (2014).
    b. Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025* (2015).
    c. Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.
    d. Sennrich, Rico, and Barry Haddow. "Linguistic input features improve neural machine translation." *arXiv preprint arXiv:1606.02892* (2016)
    e. Papineni, Kishore, et al. "BLEU: a method for automatic evaluation of machine translation." *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002.

Appendix A: List of Services

A. Tokenization

In lexical analysis, **tokenization** is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing such as parsing or text mining. Tokenization is useful both in linguistics (where it is a form of text segmentation), and in computer science, where it forms part of lexical analysis.

Our pipeline will deploy APIs for Normal text tokenization and Social media text tokenization for english language. It is so becau

B. Named Entity Recognition (NER)

Named-entity recognition is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into predefined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.

C. Transliteration

While dealing with code mixed data and cross lingual setting, transliteration is a commonly used tool. We use the character to sound mapping for transliteration.

D. Representation : In any ML application, including NLP, we must find a way to represent our data (a series of texts) to our systems (e.g. a text classifier), as we know computers only understand numbers. Word2vec, BERT, ELMO are examples of such representations.

In this project BERT as a service will be deployed.BERT, which stands for Bidirectional Encoder Representations from Transformers, is a neural network-based technique for natural language processing pre-training. In plain English, it can be used to help Google better discern the context of words in search queries.

E. Sentiment Analysis

Sentiment Analysis (SA) is the interpretation and classification of sentiment (positive, negative and neutral) within text data using text analysis techniques. A sentiment analysis system for text analysis combines natural language processing (NLP) and machine learning techniques to assign weighted sentiment scores to the entities, topics, themes and categories within a sentence or phrase.

Our sentiment analysis model detects polarity score within given text, based on this polarity score model assigns a sentiment label. Here for our model used the coarse-grained sentiment analysis labels those are Positive, Negative, Neutral. In this NLP-as-a-toolkit service we are building the SA model for normal English text

and Hinglish (Hindi and English Code-Mixed data) text. To build the model for normal English text used the open-source libraries Textblob and MIT's VADER (Valence Aware Dictionary and sEntiment Reasoner) and for Code-Mixed text used the Multi Layer Perceptron with TFIDF vectors.

F. Summarization

Text summarization refers to the technique of shortening long pieces of text. The intention is to create a coherent and fluent summary having only the main points outlined in the document.

G. Machine Translation system

Machine Translation system is a program/model for translating from source language (e.g. English) to target language (e.g. Hindi). There are multiple ways of making such a system e.g. rule based, data driven techniques like ML algorithms etc. We are going to use ML based models for building such a system.

a. Input to MT system

i. Source text (language to be translated)
ii. Language (target language) to which source has to be translated to.
iii. Reference (if user has reference - the correct translation in target language and she/he wants to find a score for translation)
iv. Which MT system to use from options given(there will be multiple MT systems which are going to be available).

b. Preprocessing

i. Purpose: Make text usable for processing by ML algorithms.
ii. Input: text(lines/File) from User
iii. Process:
    1. in house (in house means that a tokenization algorithm present in MT system as service) tokenization scheme
    2. or using tokenization service
iv. Output: sentence segmented, tokenized and normalized sentences.

c. Linguistic Features engineering (optional)

i. Purpose: Getting linguistic features for source text incase User wants to use the MT system which makes use of linguistic features.
ii. Input: line for which we need linguistic features, kind of features required (POS tagging, dependency parsing, NER's)
iii. Process:
    1. in house feature generators

2. EXTENSION: Can include features from other APIs provided they adhere to the same schema to generate features as in house feature generators.

iv. Output: sentence along with features for each token.

d. MT system_i (using system_i based on the target language and MT system technique to be used)

i. Purpose: Translating sentence from Source to Target Language.
ii. Input: taking input line by line and translating.
iii. Output: giving translation output in text/file format

e. MT Evaluation based on output from MT System:

i. Text or file containing translation of source text in target language.
ii. Text or file containing reference translation of source text in target language.
iii. Based on User Preference given at the time of input, output is sent to the Evaluation Module for getting a score of translation.
iv. Send translation back to the landing page optionally with score.