Aligning facts stored in Wikidata to sentences present in Hindi Wikipedia

Project Deliverable-1 Report

Shivprasad Sagare, Swayatta Daw
Meghana Bommadi, Vijay Vardhan Alluri

Mentor: Tushar Abhishek

15th March 2021

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Contents

1 Introduction

1.1 Scope

The problem statement we are working on is ’Aligning facts stored in Wikidata to sentences present in Hindi Wikipedia.’ Alignment of wiki data facts (in English) to sentences in different languages will help obtain more accurate language labels for Wikidata entries making Wikidata entries more multi-lingual. As explained earlier in the problem outline document, our project’s scope is to build a model that retrieves the triples that correspond to the queried sentence.

1.2 First Deliverable

This report presents the work done as part of deliverable 1, due on March 15th.

- The work includes preparing the data consisting of Wikidata facts(also called triples) and Hindi Wikipedia article text for several entities. The whole process for fetching the data from wiki data and MediaWiki APIs and preprocessing is explained in the section 2.1.

- The data fetched was found to be highly non-uniform in terms of frequency and length of sentences as well as triples. We performed an exploratory data analysis to visualize the central tendencies, spread the data, and remove the outliers to make it more uniform. Section 2.2 illustrates this in detail.
- Section 3 explains the process we applied to prepare the test data, which is used to evaluate and compare the performance of models we experiment with within the course of this project. Also, the metric used for evaluation is explained further.
- Once we had our data and evaluation strategy ready, we started with the methods proposed in the problem outline document. We have built two baseline methods, the word similarity approach, vector similarity approach, as part of this deliverable. Section 4 illustrates these methods and results of these models with the given evaluation metric strategy.
- Section 5 describes the conclusions and the plan for the next phase that is the future scope of the system.

2 Data

Our dataset includes, for each chosen entity, English Wikidata triples, and Hindi Wikipedia article sentences. Based on our survey of the categories of articles present on Wikipedia, we have decided to work on the ‘Person’ domain.

2.1 Data fetching and preprocessing

Based on our survey of the categories of articles present on Wikipedia, we have chosen to work on the ‘Person’ domain.

Sub-domain	Total entities	Fetched successfully	Failed fetching
cricketers	2597	2596	1
actors	2504	2503	1
politicians	6337	6337	0

Table 1: Statistics of Wikidata triples fetched using Python SPARQL query API

2.1.1 Obtaining the triples

We extracted the articles from multiple sub-domains using Hindi site links present in Wikidata for each entity and the triples using SPARQL query on that entity. The triples are stored as JSON mapping with the Qid of that entity as a unique top level identifier.

2.1.2 Obtaining the textual content

After obtaining the triples, we have fetched the text content from Wikipedia articles and added it to the corresponding JSON objects. We have used the Hindi Wikipedia API to obtain the content of the article. We have used the sentence_tokenize() module from indic-nlp-library to tokenize the sentences from the article. Once the data is parsed, the text content would be appended to the same JSON structure.

2.2 Exploratory Data Analysis

The fetched data consisted of entities with their corresponding triples and sentences. As it was fetched directly from the web, we performed an exploratory data analysis (EDA) to get insights into the data. We analyzed parameters like length of sentences, frequency of a specific triple across total entities, number of a sentence per entity, and number of triples per entity.

After observing the central tendencies and spread of the above parameters, we decided to remove the data points with extreme or outlier values for the above parameters. Finally, we present our final dataset, which consists of more uniform values for the above parameters. A detailed analysis of the data concerning these four parameters is shown below.

2.2.1 Frequency of a predicate across all entities

In the data fetched, there are triples with predicates that occur very less number of times. These triples won't be effective in our final output. We plot the count of triples vs their frequency across whole dataset. Data is analyzed separately for each sub-domain i.e. cricketers, actors, and politicians.

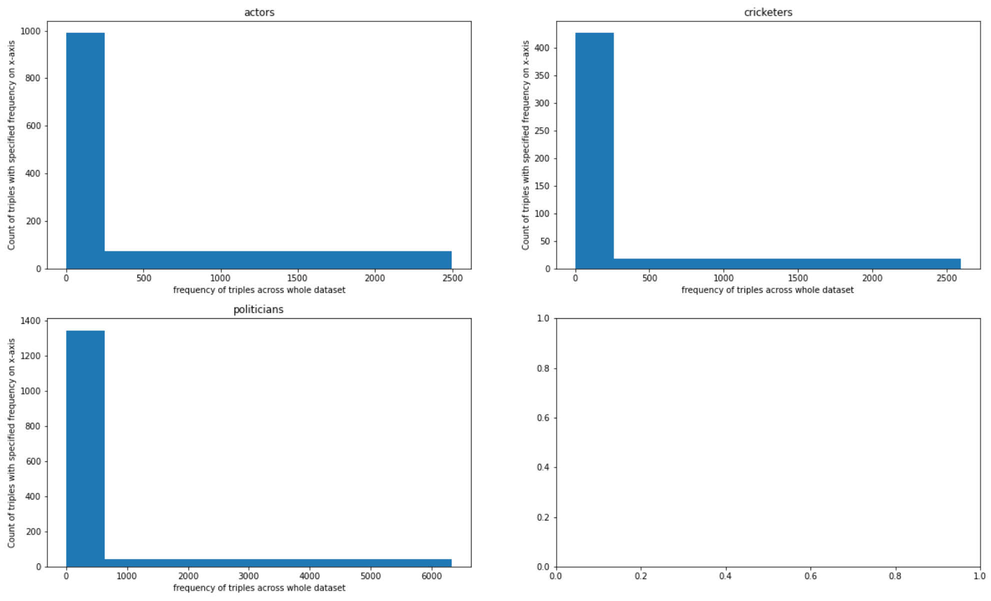


Figure 1: Distribution of the predicates based on total occurrences in the dataset. The bin partition is made at $total\_entities/10$

It can be seen in figure 1 that majority of predicates occur in less than one tenth of total entities data. Triples with such predicates are then removed from the data to include only those predicates which occur significant times. The comparison of total triple count before and after removal is given in table 2.

Sub-domain	Initial number of total triples	Reduced number of triples after deleting the predicates with less frequency
Actors	97628	70973
Cricketers	38493	34443
Politicians	171413	112284

Table 2: Total triples count for each sub-domain

2.2.2 Length of sentences

The sentences in the dataset are obtained by sentence tokenization of Hindi Wikipedia article text. Due to possible errors in tokenization, there is chance that sentences are very incorrect or consist of only phrases, titles in the text. To remove such incorrectly classified sentences, we delete those with length less than certain threshold. Also, we remove sentences with more length. Those sentences are bound to contain a lot of information and hence not suitable for mapping with triples. This was observed from our analysis of data that longer sentences often have more subjective information in them which is not present in factual triples in our dataset.

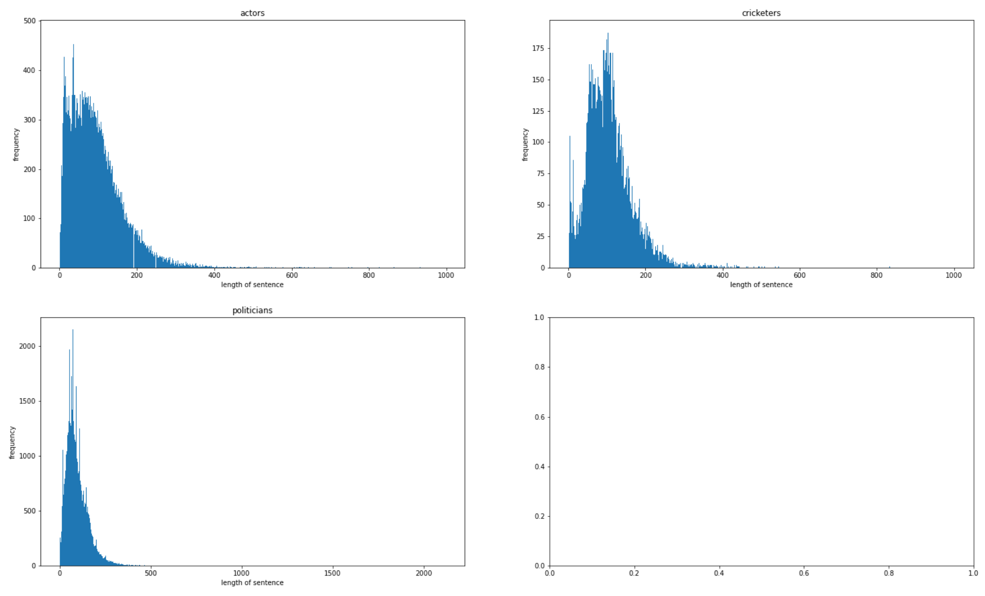


Figure 2: Distribution of the sentences based on their length. Length is measured in characters and not words.

It can be seen in figure 2 that sentence length values are roughly centered around some mean value. We remove the sentences from the dataset with length beyond 1 standard deviation from the mean observed above. This will eliminate the sentences which are either too small or too large. The detailed measures of mean, standard deviation and number of sentences is given in table 3.

Sub-domain	initial data obtained from previous step			after removing sentences with length beyond 1 standard deviation from mean		
	mean	std dev	#sentences	mean	std dev	#sentences
actors	94	69	51390	83	37	38085
cricketers	106	60	19362	98	31	14720
politicians	95	63	78308	84	33	60055

Table 3: Statistical analysis showing the mean length of sentences and std deviation along with total number of sentences available for each sub-domain.

2.2.3 Number of triples per entity

There are certain entities which have very less triples associated to them. In such case, there are less corresponding mappings in sentence space. Also, such entities won't add much information in our output dataset. So we remove the entities which have relatively less number of triples. We also remove extreme values on higher end.

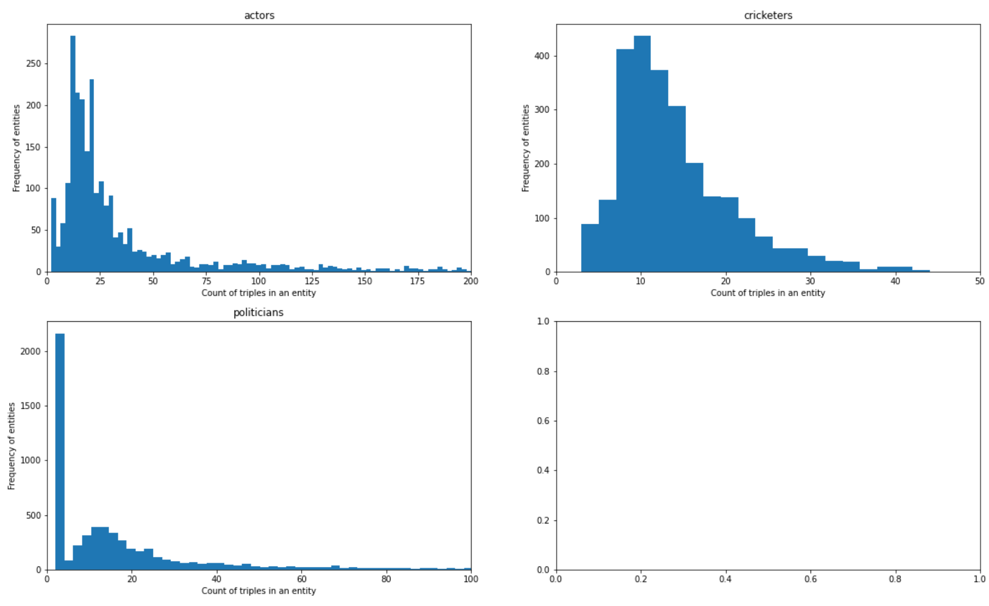


Figure 3: Plot showing distribution of frequency of triples in an entity.

We can see in figure 3 that the frequency of triples per entity follows roughly a normal distribution. We decide to remove the entities which have triple count which is beyond one std deviation from mean on both sides.

2.2.4 Number of sentences per entity

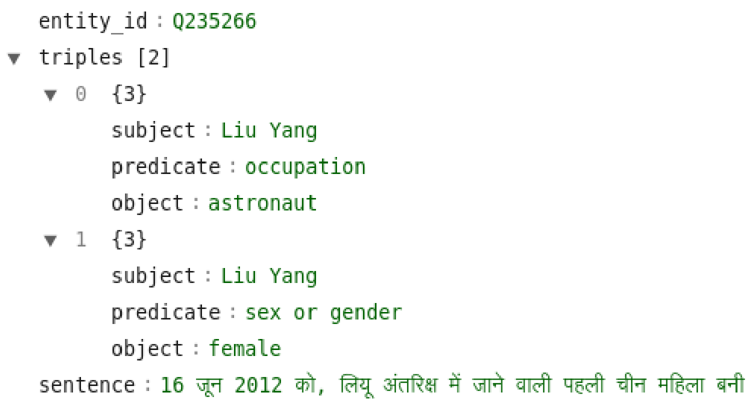
We also analyze the number of sentences present in an entity. As with previous step, the entities with relatively less number of sentences won't add much information. So we remove the sentences beyond one std deviation from mean.

The initial fetched data goes through above 4 successive transformations. The final updated dataset is available on the GitHub repository. Interested readers can also access the code for above EDA in stats.ipynb notebook available on the repository.

3 Test data & Evaluation Metrics

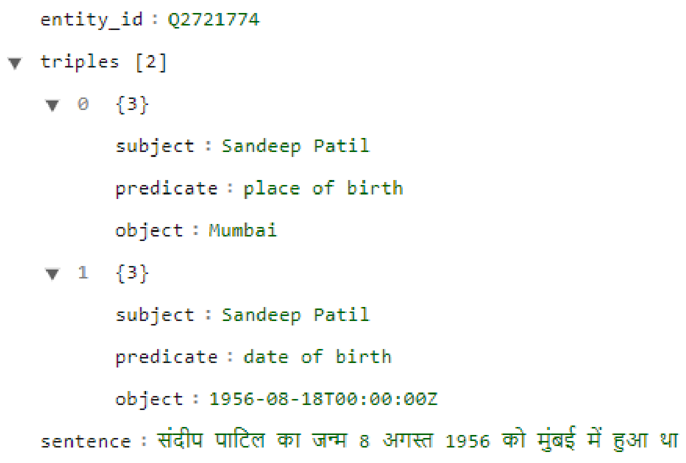
3.1 Test data preparation

For preparing test data to evaluate our methods, we manually annotate the data points by aligning the triples to sentences from the dataset obtained from above transformations. We decided to create the test data containing 400 sentences and their aligned triples. Below is a sample instance from the dataset.



```
  entity_id : Q235266
▼ triples [2]
  ▼ 0  {3}
       subject : Liu Yang
       predicate : occupation
       object : astronaut
  ▼ 1  {3}
       subject : Liu Yang
       predicate : sex or gender
       object : female
  sentence : 16 जून 2012 को, लियू अंतरिक्ष में जाने वाली पहली चीन महिला बनी
```

Figure 4: JSON structure of a sample instance in test data. This instance denotes partial coverage of information in sentence by triples.



```
  entity_id : Q2721774
▼ triples [2]
  ▼ 0  {3}
       subject : Sandeep Patil
       predicate : place of birth
       object : Mumbai
  ▼ 1  {3}
       subject : Sandeep Patil
       predicate : date of birth
       object : 1956-08-18T00:00:00Z
  sentence : संदीप पाटिल का जन्म 8 अगस्त 1956 को मुंबई में हुआ था
```

Figure 5: This instance denotes full coverage of information in sentence by triples.

As this process of creating the test data needed a lot of manual efforts, we planned on building a web application in order to help us annotate the data faster. The web application would give us the JSON object from our initial dataset which contain the triples and sentence. Based on the sentence, we could select the relevant triples from a dropdown/checkbox. After selecting the triples and hitting save, the web application would automatically save the sentence and triples and store them in JSON format.

3.2 Evaluation Metrics

The output of our alignment model is a set of triples corresponding to the queried sentence. We decide to use the evaluation metrics of precision and recall to compare the predicted triples with those present in test data. Precision denotes ratio of the number of predicted triples which are actually correct mappings to the number of all triples predicted. Recall signifies ratio of the number of correctly predicted triples to the number of all the actual correct triples corresponding to that sentence.

4 Experiments & Results

As our baseline model, we are using static multilingual embeddings to align the English triples and Hindi sentences in common vector space. We use MUSE from Facebook Research for this. MUSE is a Python library for multilingual word embeddings, whose goal is to provide the community with: state-of-the-art multilingual word embeddings (fastText embeddings aligned in a common space) and a large-scale high-quality bilingual dictionaries for training and evaluation.

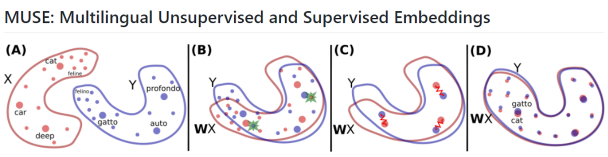


Figure 6: MUSE Multilingual Embeddings

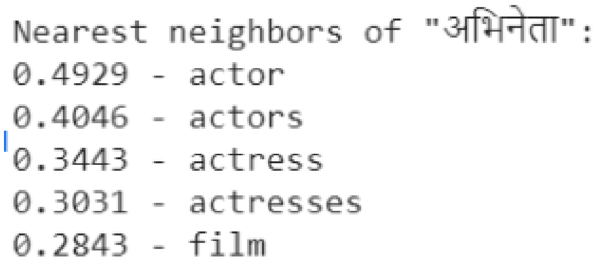
We have used :

- MUSE pre-aligned (En-Hi) embeddings made available by Facebook.
- We also train our own alignments(En-Hi) using MUSE using a train bilingual dictionary (or identical character strings as anchor points), to learn a mapping from the source to the target space.

The baseline approaches we developed as a part of this deliverable are explained below.

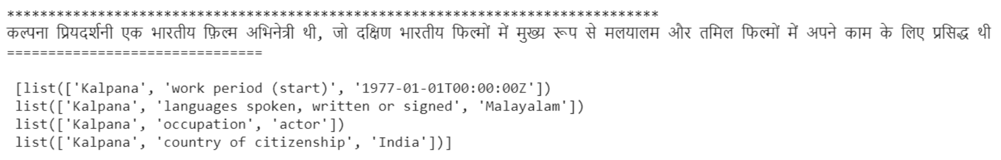
4.1 Word Overlap

For this method, we have used the MUSE pre-trained aligned multilingual embeddings made publicly available by Facebook. We obtain the multilingual word embeddings in Hindi for every word in Hindi sentences. Then, using k-nearest neighbours algorithm(kNN), we find the most semantically similar English words to these Hindi words. We use k = 5 to get 5 most similar English words for each word in the Hindi sentence. For out of vocab words we use google translate for direct transliteration. We keep all these words in a list and then compare these English words with the words in entity triples consisting of both predicate and object. We obtain a score for the amount of overlap of the words with each triple. We set a threshold score, and return the triples which have a greater score than the threshold.



```
Nearest neighbors of "अभिनेता":
0.4929 - actor
0.4046 - actors
0.3443 - actress
0.3031 - actresses
0.2843 - film
```

Figure 7: Top 5 similar words are used for finding the word overlap. This improves the flexibility as compared with direct string matching of only one particular word.



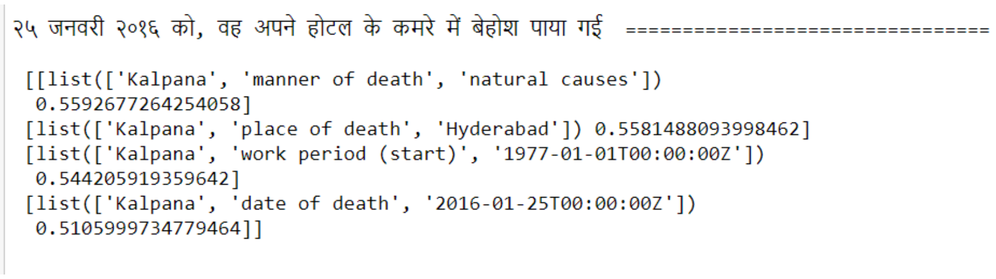
```
********************************************************************************
कल्पना प्रियदर्शनी एक भारतीय फ़िल्म अभिनेत्री थी, जो दक्षिण भारतीय फिल्मों में मुख्य रूप से मलयालम और तमिल फिल्मों में अपने काम के लिए प्रसिद्ध थी
================================

 [list(['Kalpana', 'work period (start)', '1977-01-01T00:00:00Z'])
 list(['Kalpana', 'languages spoken, written or signed', 'Malayalam'])
 list(['Kalpana', 'occupation', 'actor'])
 list(['Kalpana', 'country of citizenship', 'India'])]
```

Figure 8: Resultant triples obtained by word overlap method on a sample sentence. Triples with score above threshold of 1 are only selected.

4.2 Vector similarity

For this method, we have used our own trained aligned Multilingual embeddings using the MUSE bilingual dictionary. Using these, we obtain word embeddings for every word in a Hindi sentence. To get sentence level embeddings, we simply take the average of the embeddings of all the words in the given sentence. For the triples, we average the word embeddings for every word in the triple. Then we find the cosine similarity between the sentences and the triples for each entity. We set a threshold of 0.5 and return the triples which have a greater score than the threshold, for each sentence.



```
२५ जनवरी २०१६ को, वह अपने होटल के कमरे में बेहोश पाया गई  ================================

 [[list(['Kalpana', 'manner of death', 'natural causes'])
  0.5592677264254058]
 [list(['Kalpana', 'place of death', 'Hyderabad']) 0.5581488093998462]
 [list(['Kalpana', 'work period (start)', '1977-01-01T00:00:00Z'])
  0.544205919359642]
 [list(['Kalpana', 'date of death', '2016-01-25T00:00:00Z'])
  0.5105999734779464]]
```

Figure 9: Resultant triples obtained by vector similarity method on a sample sentence. Triples with score above threshold of 0.5 are only selected.

4.3 Results

	Word Overlap		Vector Similarity	
Sub-domain	Precision	Recall	Precision	Recall
actors	0.432	0.825	0.321	0.508
cricketers	0.385	0.821	0.563	0.608
politicians	0.378	0.606	0.322	0.263
Average	0.3985	0.75	0.3825	0.48

Table 4: Precision and recall values corresponding to each sub-domain for both methods.

5 Conclusion & Future scope

As outlined in the scope document at the start of the project, as part of the deliverable 1, we worked upon the tasks of data processing, preparation of test data, developing baseline methods, and their evaluation. In this report, we present the methodology, challenges faced, and the solutions implemented in each of the above tasks. The future direction involves experimenting with the approaches involving transformer based multilingual embeddings and making use of TRex dataset to align the English and Hindi sentences and using pre-existing English to English mappings. We also plan to annotate more test data in coming phase for better generalization.