Text Generation from Knowledge Graphs like Wikidata

Aditya Agarwal 20161104

Aniruddha Deshpande 20161058

April 29, 2020

Abstract

This Report summarises the work done on Text Generation from Knowledge Graphs like Wikidata. Here, we touch upon various aspects such as Studying Wikidata, choosing a domain, writing code to convert features into Human Readable Format, writing template sentences, training an NLG, and finally using certain Tests to test our Page that is created.

1 Introduction

This project of Automatic text generation from Knowledge graphs aims to study, capture and convert the information stored in the form of Wikidata Knowledge graphs to that of **Hindi Wikipedia pages**. The motivation for this project comes especially due to the disappointingly less number of Wikipedia pages in Indian Languages. Increases in computing power and model capacity have made it possible to generate mostly grammatical sentence length strings of natural language text. However, generating several sentences related to a topic and which display overall coherence and discourse-relatedness is an open challenge. The difficulties are compounded in domains of interest such as scientific writing. Our aim is to generate these Wikipedia pages as human-like as possible. Unlike the already existing LSJbot, our project also aims at generating larger documents (At least 500 Words) including all the relevant information. LSJbot failed in doing so as it generated several documents spanning only upto one or two lines.

2 Language and Domain Choice

For this project, the Indian Language we are going forward with is Hindi. For the purpose of this project, the expected number of generated documents lie in the range of at least upto 10,000 documents with a minimum of 500 words per article. Keeping that in mind, we initially started working using **Monuments** as our Domain to extract information from it and later we moved on to **Films**.

3 Development Stages

3.1 Stage 1: Studied Wikidata and learnt about it's structure

3.2 Stage 2: Chose a domain to work so that we could create our Hindi Wikipedia Pages in that and retrieve all the available data from Wikidata in that domain so that we can train on that data and then test on new data

3.3 Stage 3: Converted the data downloaded into Human Readable Format to get it into the format required for training. We also finally converted the code into the box format needed for NLG training. To run the NLG, we needed the Wikipedia Pages Content from this dump and then the labels that we would convert in this step as the two inputs to the NLG

3.4 Stage 4: Created a NLG code for training purposes

3.5 Stage 5: Created template sentences so that we could use both to make the page output better.

3.6 Stage 6: Finally, testing using standardized scores and improving our final pages created.

4 Detailed Solution Outline

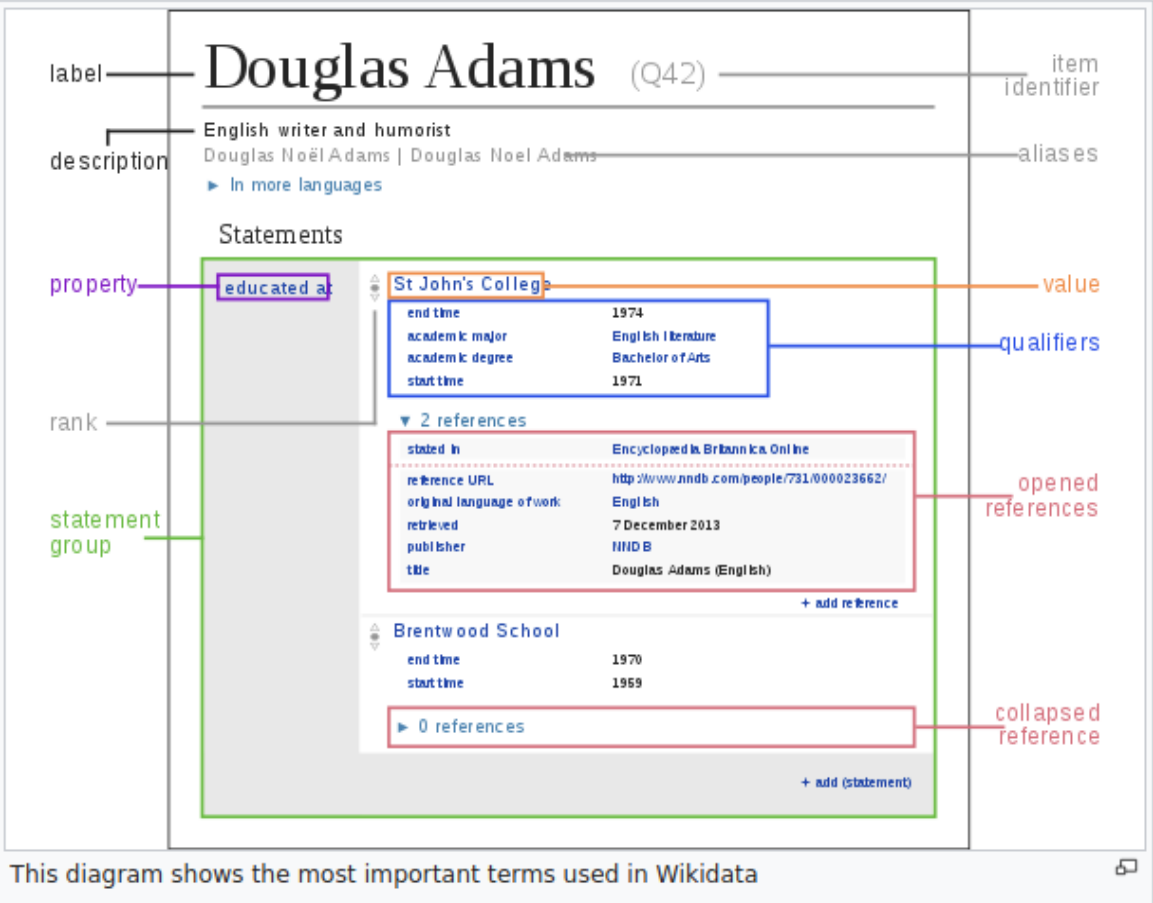
4.1 Stage 1: What is Wikidata and how it stores data?

Wikidata is a collaboratively edited knowledge base hosted by the Wikimedia Foundation. It is a common source of open data that Wikimedia projects such as Wikipedia can use, and anyone else, under a public domain license. The primary data storage is JSON blobs in an SQL database. The data in wikidata is stored in the form of specific IDs that forms the base of Wikidata. Each Wikidata entity is identified by an entity ID, which is a number prefixed by a letter. Items are prefixed with Q (e.g. Albert Einstein (Q937)), Properties are prefixed by P (e.g. instance of (P31)) and lexemes are prefixed by L (e.g. L1). The Wikidata Query Service is used to run queries on Wikidata and uses an RDF triple store to allow SPARQL queries against the current version of the data.

4.2 Stage 2: Choosing a Domain to Work and creating the Dump

Our decision of choosing Monuments as a Domain, was based on the results that we obtained from looking at the results of the SPARQL queries. We then dumped all available English Monument Data by scraping Wikidata into a JSON file and we found that we had around **24000 monuments, but out of those only 11524 monuments had English Labels**. We then checked each page with the Hindi Wikipedia and found out that only **284** out of these had Hindi Wikipedia Pages. We worked on Monuments for a bit but soon realized that the data was inconsistent and very less to train our NLG. We then switched to Films as our Primary Domain because this had **2 Lakh English Film Data and 5000 Hindi Film**

Data which was much better than the 284 Hindi Monuments Data that we had initially found.



4.3 Stage 3: Converting Dump Data into Human Readable Format

This was the biggest step that took away most of our time. We had to write multiple code files to get the data into the format we needed to train the NLG. A brief description of each code file in logical order follows:

4.3.1 WikidataDownload.ipynb

This Code was to get the Monuments and later Films Data as a JSON dump from Wikidata(which we had already downloaded by writing a SPARQL query). This code also retrieved all the English Labelled Monuments from the dump. We got 11524 English Monuments as our output and 4000 Hindi Films out of 2 Lakh Films.

4.3.2 Extract_English_WikiPages.ipynb and Extract_Hindi_WikiPages.ipynb

These codes were to extract the Respective Languages Wikipedia page from the Monument and Film Data that we got from the above code. Thus using certain libraries we were able

to get 2000 English Monuments and 222 Hindi Monuments that had Wikipedia Pages out of 11000 and 284 Original English and Hindi Monuments and 4000 Hindi Films that had Wikipedia Pages out of 5000 original Hindi Films. We needed that data that included only Wikipedia Pages because of training purposes.

4.3.3 Labelling_IDs_English.ipynb and Labelling_IDs_Hindi.ipynb

The NLG code that we would be using to train our model, required that the data would be in a certain format called a box format. So to acheive this, from the Wikidata content of the Wikipedia Pages of the Monuments, we got from the output of the previous code, we had to extract all the important properties and their values. To do this, we had to extensively look at how Wikidata stored the data, and we found out that it stored content in form of ID's which were of no use to us in that format. So we initially ran 6 nested iterations to convert each property, value pair into Human Readable Format which we then optimized to only 2 iterations to convert only useful information. After doing this, this code stored each monument with these important property value pairs in Human Readable Format. We then ran the same code for films once we decided to change the domain. Now, to ensure we get a better output from the NLG, we were advised by the mentors to send in Hindi Labels as input to the NLG but we only had Human Readable English Labels, so we had to run a Google Translate API to convert all these labels to Hindi but the Translate API gave us a lot of problems so we had no choice but to convert it ourselves manually. We manually converted around **60000 unique labels** and finally we had converted each of the 4000 films Wikidata into Hindi labels Data.

4.3.4 box_formatter.py, Data_Formatting_English.ipynb, Data_Formatting_Hindi.ipynb

These final python codes were used to transform the labels that we had translated into the box format that were required for the NLG training model. An example of how the box format looked:

We had initially all these labels such as image, coordinate location as dictionaries of values or just strings of data and now the box format required subscripts of all information in one line and so on.

Thus after running these various codes listed above, we were finally able to get the labels we wanted in the right format and sent the Wikipedia Pages and the labels to the NLG for training.

```
image_1_1:D.    image_1_2:Pedro image_1_3:IV    image_1_4:Porto.JPG     country_1:Portugal
coordinate_location_altitude_1:None     coordinate_location_precision_1:0.01    coordinate_loca
depicts_1:horse depicts_1:equestrianism instance_of_1:monument  instance_of_1:statue    instanc
located_in_the_administrative_territorial_entity_1_1:Cedofeita, located_in_the_administrative_t
located_in_the_administrative_territorial_entity_1_4:Sé,        located_in_the_administrative_t
located_in_the_administrative_territorial_entity_1_7:Nicolau    located_in_the_administrative_t
creator_1_1:Célestin    creator_1_2:Anatole     creator_1_3:Calmels     Commons_category_1_1:Es
Commons_category_1_5:IV Commons_category_1_6:na Commons_category_1_7:Praça      Commons_categor
heritage_designation_1_2:Cultural       heritage_designation_1_3:Heritage       heritage_design
DGPC_ID_1:73484 SIPA_ID_1:5572  location_1_1:Santo      location_1_2:Ildefonso  located_on_stre
height_amount_1:+10     height_unit_1:http://www.wikidata.org/entity/Q11573     material_used_1
Commons_category_1_1:Lion       Commons_category_1_2:of Commons_category_1_3:Waterloo   coordin
coordinate_location_altitude_1:None     coordinate_location_precision_1:1e-05   coordinate_loca
Waterloo-Butte-du-Lion-statue.jpg       country_1:Belgium       located_in_the_administrative_t
architect_1_3:Straeten  instance_of_1:monument
```

4.4 Stage 4: NLG and Template Sentences

Our model is based on the paper 'Order-Planning Neural Text Generation From Structured Data' and takes as input a table (e.g., a Wikipedia infobox) and generates a natural language summary describing the information based on an RNN.

Table:

ID	Field	Content
1	Name	*Arthur Ignatius Conan Doyle*
2	Born	*22 May 1859 Edinburgh, Scotland*
3	Died	*7 July 1930 (aged 71) Crowborough, England*
4	Occupation	*Author, writer, physician*
5	Nationality	*British*
6	Alma mater	*University of Edinburgh Medical School*
7	Genre	*Detective fiction fantasy*
8	Notable work	*Stories of Sherlock Homes*

Text:

Sir Arthur Ignatius Conan Doyle (22 May 1859 – 7 July 1930) was a British writer best known for his detective fiction featuring the character Sherlock Holmes.

This paper presents a new approach to Natural Language Generation as a neural network should model not only word order (as has been well captured by RNN) but also the order of contents, i.e., fields in a table. We also observe from real summaries that table fields by themselves provide illuminating clues and constraints of text generation. In the biog-

raphy domain, for example, the nationality of a person is typically mentioned before the occupation. This could benefit from explicit planning of content order during neural text generation The neural network contains three main components:

4.4.1 Encoder: This captures table information

The content of each field is split into separate words and the entire table is transformed into a large sequence. Then we use a recurrent neural network(RNN) with long short term memory (LSTM) unit to read the contents as well as their corresponding field names. Notice that, we have two separate embedding matrices for fields and content words. We observe the field names of different data samples mostly come from a fixed set of candidates, which is reasonable in a particular domain. Therefore, we assign an embedding to a field, regardless of the number of words in the field name.

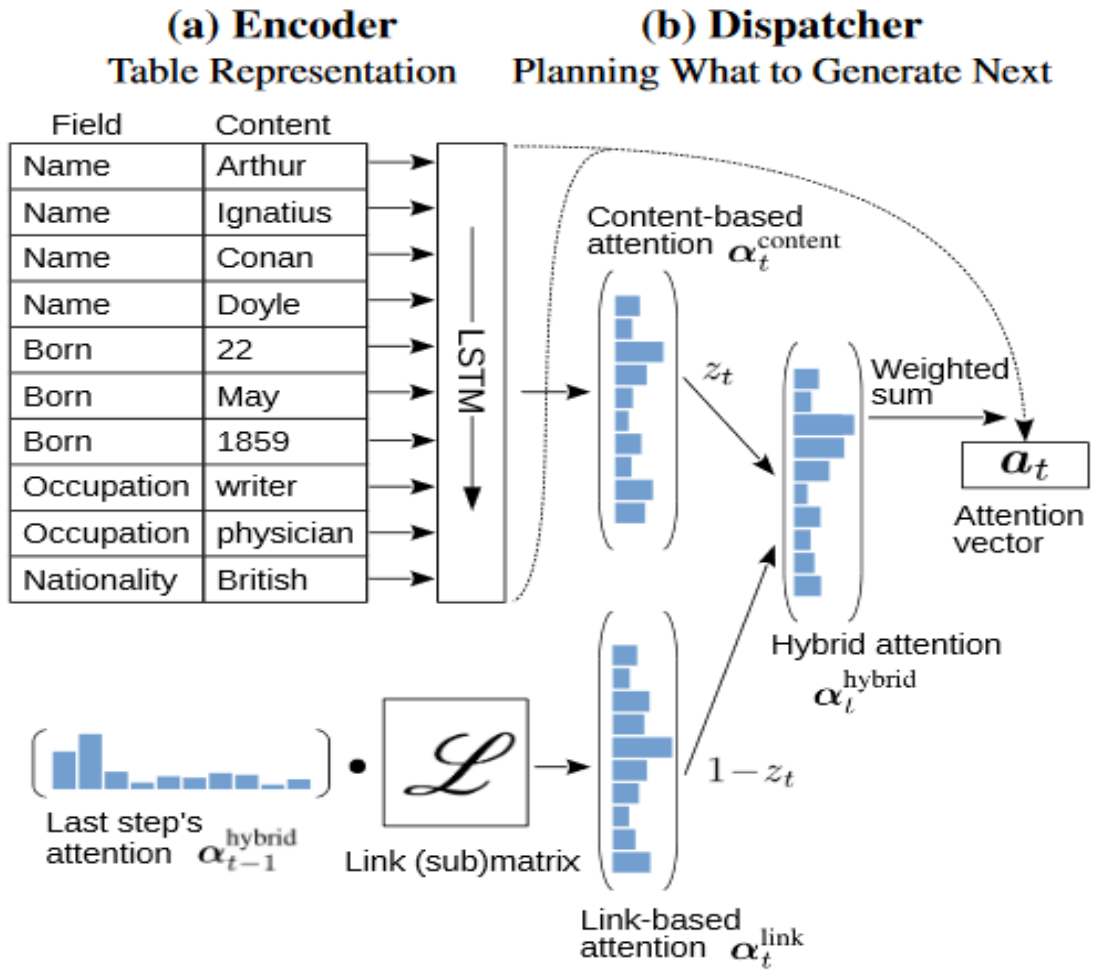


Figure 1: The (a) Encoder and (b) Dispatcher in our model.

4.4.2 Dispatcher: A hybrid content and linkage-based attention mechanism over table contents, plans what to generate next

Generally, a dispatcher is an attention mechanism over table contents. At each decoding time step t, the dispatcher computes a probabilistic distribution which is further used for weighting content representations. It uses a hybrid of Content-Based Attention and Link-Based Attention.

4.4.2.1 Content-Based attention

Since both the field name and the content contain important clues for text generation, we compute the attention weights based on not only the encoded vector of table content but also the field embedding, thus obtaining the final attention by re-weighting one with the other.

4.4.2.2 Link-Based attention

We further propose a link-based attention mechanism that directly models the relationship between different fields. Our intuition stems from the observation that, a well-organized text typically has a reasonable order of its con-tents. As illustrated previously, the nationality of a person is often mentioned before his occupation (e.g.,a British writer). Therefore, we propose an link-based attention to explicitly model such order information.

4.4.2.3 Hybrid attention

To combine the above two attention mechanisms, we use a self-adaptive gate zt (0,1) by a sigmoid unit.

4.4.3 Decoder: Generates a natural language summary using RNN, where we also incorporate a copy mechanism to cope with rare words

The decoder is an LSTM-RNN that predicts target words in sequence. We also have an attention mechanism that summarizes source information,i.e., the table in our scenario, by weighted sum, yielding an attention vector where hi is the hidden representation obtained by the table encoder. As is a probabilistic distribution—determined by both content and link information—over content words, it enables the decoder RNN to focus on relevant information at a time, serving as an order-planning mechanism for table-to-text generation.

Thus, this approach was used to train our data that we made using the codes above.

One of the main reasons of changing our domain was based on the output we got from this code. Using monuments, we found out that the output sentences that we were getting were not making sense, as firstly the dataset was extremely small and secondly, the data was incoherent in the first place. Thus, we switched to Films as our domain. This resulted in much better results and a better accuracy.

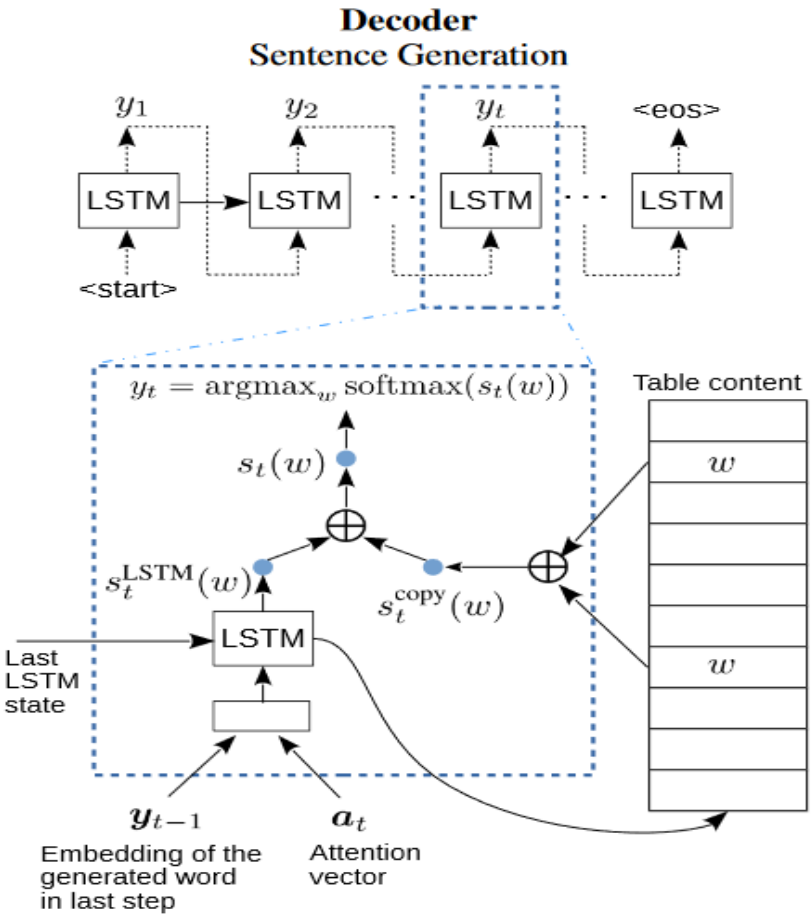


Figure 2: The decoder RNN in our model, which is enhanced with a copy mechanism.

5 Template Sentences Generation

This was the most recent development we did since the presentation and we got the best results from this. To generate Template Sentences, we started with reviewing all the properties that we had got, and compared those properties with each already existing Hindi Wikipedia Page about Films. To enable us to create logical and meaningful template sentences, we looked for each of these properties in atleast 100 Wikipedia Pages on Films in Hindi and if they existed in atleast 100, we assumed that these properties should be there for every film. We then edited out some of the properties that we thought would not be helpful to describe a film and were finally left with some properties using which we created template sentences. A very important point that we followed while using these template sentences to display the Wikipedia page was to use **Consecutive Shortening of Use of Properties in every Consecutive Sentence.** This meant, that we initially generated simple sentences with one label, but soon we were generating complex sentences with multiple labels also. Looking at various Wikipedia pages we found out our starting sentence and hence figured that Complexity of the sentence and the order content couldn't go hand in

hand because we took the complex sentences having many labels and printed it first, and then removed those properties from the list and all sentences pertaining to those. Finally from the remaining properties we again made the sentences and the process carried on until no property remained.

```
template_sentences = ['{{film}} एक {{based on}} पर आधारित फिल्म है। ',
                      'इस फिल्म का  निर्देशन {{director}} ने किया है। ',
                      ' यह फिल्म का वितरण {{distributor}} द्वारा किया गया है। ',
                      ' फिल्म की घटनाएं {{narr locations}} पर हुई हैं। ',
                      'फिल्म के लिए संगीत रचना {{music director}} द्वारा दे गयी है और गायन {{singers}} द्वा
रा दिया गया है। ',
                      'फिल्म के लिए संगीत रचना {{music director}} द्वारा दे गयी है। ',
                      'यह फिल्म {{artist}} द्वारा निर्माणित कथा पर आधारित है। ',
                      '{{film}} {{release year}} में बनी {{genre}} शैली की फ़िल्म है जिसका निर्माण {{arti
st}} ने किया है व वितरण {{distributor}} द्वारा किया गया है। ',
                      'इस फिल्म की घटनाएं {{narr locations}} पर हुई गयी थी लेकिन इनकी शूटिंग {{fil loca
tions}} पर की गयी थी। ',
                      'फिल्म  की शूटिंग {{fil locations}} पर हुई हैं। ',
                      '{{film}} एक {{based on}} पर आधारित {{genre}} शैली की फिल्म है।',
                     '{{film}} एक {{based on}} पर आधारित {{topic}} के विषय पर बनी फिल्म है।',
                     '{{film}} एक {{based on}} पर आधारित फिल्म है जो {{series}} शृंखला का हिस्सा है
।',
                     '{{film}} एक {{based on}} पर आधारित {{genre}} शैली की फिल्म है और {{series}}
शृंखला का हिस्सा है ।',
                      'यह फिल्म {{based on}} पर आधारित {{genre}} शैली की फिल्म है। ',
                      'यह फिल्म {{based on}} पर आधारित {{topic}} के विषय पे बनी फिल्म है । ',
                      'यह फिल्म {{based on}} पर आधारित {{series}} शृंखला का हिस्सा है । ',
                      'यह फिल्म {{based on}} पर आधारित {{genre}} शैली की फिल्म है और {{series}} शृंख
ला का हिस्सा है ।',
                      '{{film}} एक {{release year}} में रिलीज़ हुई {{genre}} शैली की फिल्म है। ',
                      '{{film}} एक {{release year}} में रिलीज़ हुई {{topic}} के विषय पर बनी फिल्म है ।',
                      '{{film}} {{release year}} में रिलीज़ हुई, एक {{series}} शृंखला की फिल्म है ।',
                      '{{film}} एक {{series}} शृंखला की फिल्म है ।',
                      '{{film}} एक {{release year}} में रिलीज़ हुई, {{artist}} द्वारा निर्माणित कथा पर आधा
रित फिल्म है ।',
                      '{{film}} एक {{release year}} में रिलीज़ हुई, {{artist}} द्वारा निर्माणित कथा पर आधा
रित {{series}} शृंखला की फिल्म है।',
                      '{{film}} एक {{release year}} में रिलीज़ हुई, {{topic}} के विषय पर बनी {{serie
s}} शृंखला की फिल्म है।',
                      '{{film}} एक {{release year}} में रिलीज़ हुई, {{genre}} शैली की फिल्म है जो एक
{{series}} शृंखला का हिस्सा है । ',
                      'इस फिल्म में {{singers}} ने गायन किया है।',
```

This had a negative consequence that the order was not maintained. For example, Consider a case where in for the film, both property about director and producer of the film existed. Here the template sentence generator will generate a sentence containing both these properties first as it has a higher complexity since it contains 2 properties over say a sentence about a single property of the costume designer of the film. Now consider a case wherein the producer property exists but director property doesn't exist and the costume designer property exists. Now while generating template based sentences here, there are sentences with only singular properties i.e. producer and costume designer. In ideal cases producer is mentioned first in the article but our code of generating template sentences gives equal weights to such singular properties and hence cases of the costume designer being mentioned before the producer can occur as they are treated equal by our code.

We tried to minimise these ordering issues by carefully selecting related and more useful properties and generating as complex sentence as it can generate and include the higher

priority properties in the complex sentence constructions. We did this to minimise the above mentioned ordering issues.

'जुरासिक पार्क एक 1993 में रिलीज़ हुई, विज्ञान कथा फिल्म शैली की फिल्म है जो एक जुरासिक पार्क शृंखला का हिस्सा है । कैथलीन कैनेडी, गेराल्ड आर मोलेन, स्टीवन स्पीलबर्ग द्वारा निर्मित इस फिल्म का लेखन माइकल क्रिक्टन, डेविड कोएप्प ने किया और निर्देशन स्टीवन स्पीलबर्ग ने संम्भाला।इस फिल्म की घटनाए कोस्टा रिका, इसला नुबलर पर हुई गयी थी लेकिन इनकी शूटिंग हवाई, काउई पर की गयी थी। फिल्म की लागत 63000000 डॉलर थी और उसकी तुलना में फिल्म ने 1030314141 डॉलर कमाए।यह मौलिक रूप से अंग्रेज़ी भाषा में शूट हुई एक संयुक्त राज्य अमेरिका की फिल्म है।यह फिल्म जुरासिक पार्क पर आधारित डायनासोर के विषय पे बनी फिल्म है । इस फिल्म ने कई पुरस्कार मनोनीत किया गया जैसे सर्वश्रेष्ठ ध्वनि संपादन के लिए अकादमी पुरस्कार, सर्वश्रेष्ठ ध्वनि मिश्रण के लिए अकादमी पुरस्कार, सर्वश्रेष्ठ दृश्य प्रभाव के लिए अकादमी पुरस्कार और इन में से कई पुरस्कार जैसे सर्वश्रेष्ठ ध्वनि संपादन के लिए अकादमी पुरस्कार, सर्वश्रेष्ठ ध्वनि मिश्रण के लिए अकादमी पुरस्कार, सर्वश्रेष्ठ दृश्य प्रभाव के लिए अ कादमी पुरस्कार, राष्ट्रीय फिल्म रजिस्ट्री जीते भी।फिल्म के सम्पादक माइकल कहन थे और इनके साथ फोटोग्राफी निर्देशन डीन कुंडी ने संभाली ।रिचर्ड एटनबरो, जेफ गोल्डब्ल म, सैमुअल एल जैक्सन, सैम नील, लौरा डर्न, एरियाना रिचर्ड्स ने इस फिल्म में भूमिका निभाई । यह फिल्म का वितरण यूनिवर्सल स्टूडियोज द्वारा किया गया है। यह एक रंग फिल्म है। इस फिल्म में जॉन विलियम्स ने गायन किया है।फिल्म के लिए प्रोडक्शन डिजाइनिंग रिक कार्टर ने की ।'

Figure 1: Template Wikipage generated for Jurassic Park

After doing the above procedure, we got something like this as our final Output for the movie Jurassic Park. We could even better this output once the NLG output for the same would be combined with the above Template Wikipage Generated

6 Major Hurdles, How we solved them and Future Work

We faced many hurdles along the way, some of them include:

6.1 Conversion to Human Readable Format

To code the converted JSON dump required a lot of effort and took away most of our time as we had to iterate through multiple keys, with those having multiple subkeys and id's. Our code was heavily unoptimized and to run our code on 11000 articles would require a lot of time, thus we needed an extremely efficient method to run it as fast as possible and we did as we removed unnecessary iterations that we felt were of no use to us. Also during the entire execution, there existed some properties like P727, P1619 which had no wikidata and hence gave an error so we had to code these separately.

6.2 Lack of Data to train in case of NLG

Lack of Data was our major problem. An NLG requires a lot of data to train but due to number of Hindi Monument Wikipedia Pages being only 284, our training set was extremely small and would not help us in creating the maximum accuracy model for making Hindi Wikipedia Pages. The weights would heavily overfit and the outputs may turn out to be extremely poor as some Wiki Pages are extremely small without much content.

6.3 Problems with Translate API's

We required the translate API's because according to our mentors, the NLG would better train if we used Hindi Labels along with the Hindi Wikipedia Pages. The problem with Google Translate API was that it didn't allow so many requests at a time even with sleep, and our IP's got blocked. We tried other Translate API's but the accuracy was even worse than Google Translate. Bing Translator API had a limit of only 20 lakh characters per month which was never going to be enough for the number of labels we had. Thus finally our only solution remained, was to manually translate the English Labels to Hindi.

6.4 Errors during Training NLG on ADA

We trained our model on ADA but the training was not very nice because it kept running out of memory on ADA and the training used to stop every 12 hours. We tried reducing the batch size to 2 but it still gave out of memory errors.

The above reason is also a cause for our BLEU scores being close to zero even though the training went fine. Extensive training could solve these issues by a little bit but since it is an Indian Language NLG, it may still be a difficult task.

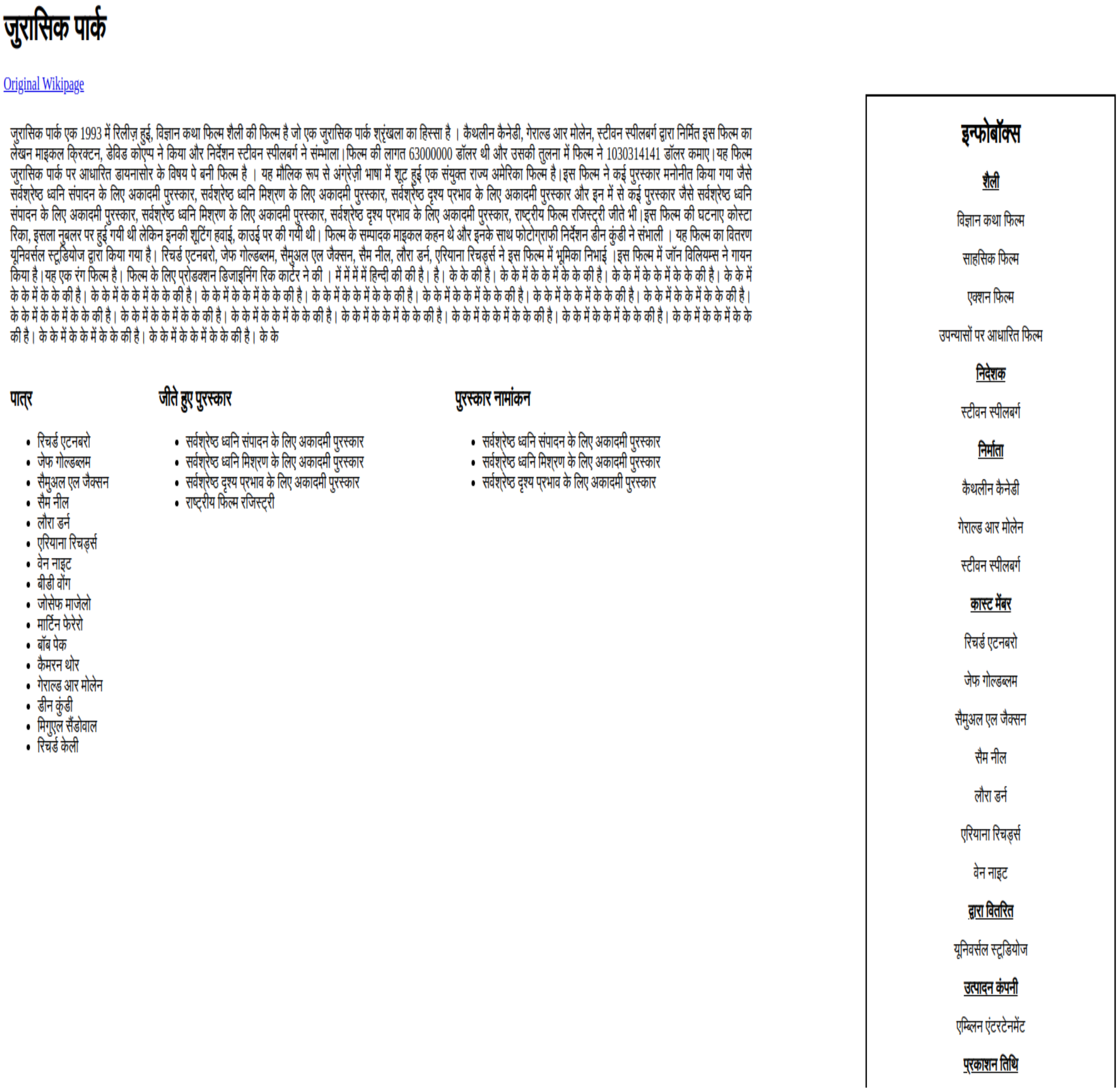
6.5 Improvisation Techniques

We can achieve better training with higher computational power or by probable extra data management.

Also, if we have more data, the NLG has more resources to run on, and hence will give us a better output.

We can also achieve better training if we actually train our NLG using the template sentences we have generated because it actually contains stuff from the Wikidata. This is because, as we saw during training NLG with just labels, that the Wikipedia Page had sections like PLOT but the wikidata dump did not and hence both had inconsistent matching and hence could not produce good results. But the template sentences could be made regarding plot and hence now we could match the exact Hindi Wikipedia Page with the template sentences.

We also have created a code that combines both the output from the NLG and the template sentences and produces a final Wikipedia Page, with an infobox also. We have been unable to add images because there is no link to the image in the wikidata dump downloaded.



जुरासिक पार्क

Original Wikipage

जुरासिक पार्क एक 1993 में रिलीज़ हुई, विज्ञान कथा फिल्म शैली की फिल्म है जो एक जुरासिक पार्क शृंखला का हिस्सा है । कैथलीन कैनेडी, गेराल्ड आर मोलेन, स्टीवन स्पीलबर्ग द्वारा निर्मित इस फिल्म का लेखन माइकल क्रिक्टन, डेविड कोएप्प ने किया और निर्देशन स्टीवन स्पीलबर्ग ने संम्भाला।फिल्म की लागत 63000000 डॉलर थी और उसकी तुलना में फिल्म ने 1030314141 डॉलर कमाए।यह फिल्म जुरासिक पार्क पर आधारित डायनासोर के विषय पे बनी फिल्म है । यह मौलिक रूप से अंग्रेज़ी भाषा में शूट हुई एक संयुक्त राज्य अमेरिका फिल्म है।इस फिल्म ने कई पुरस्कार मनोनीत किया गया जैसे सर्वश्रेष्ठ ध्वनि संपादन के लिए अकादमी पुरस्कार, सर्वश्रेष्ठ ध्वनि मिश्रण के लिए अकादमी पुरस्कार, सर्वश्रेष्ठ दृश्य प्रभाव के लिए अकादमी पुरस्कार और इन में से कई पुरस्कार जैसे सर्वश्रेष्ठ ध्वनि संपादन के लिए अकादमी पुरस्कार, सर्वश्रेष्ठ ध्वनि मिश्रण के लिए अकादमी पुरस्कार, सर्वश्रेष्ठ दृश्य प्रभाव के लिए अकादमी पुरस्कार, राष्ट्रीय फिल्म रजिस्ट्री जीते भी।इस फिल्म की घटनाए कोस्टा रिका, इसला नुबलर पर हुई गयी थी लेकिन इनकी शूटिंग हवाई, काउई पर की गयी थी। फिल्म के सम्पादक माइकल कहन थे और इनके साथ फोटोग्राफी निर्देशन डीन कुंडी ने संभाली । यह फिल्म का वितरण यूनिवर्सल स्टूडियोज द्वारा किया गया है। रिचर्ड एटनबरो, जेफ गोल्डब्लम, सैमुअल एल जैक्सन, सैम नील, लौरा डर्न, एरियाना रिचर्ड्स ने इस फिल्म में भूमिका निभाई ।इस फिल्म में जॉन विलियम्स ने गायन किया है।यह एक रंग फिल्म है। फिल्म के लिए प्रोडक्शन डिजाइनिंग रिक कार्टर ने की । में में में में हिन्दी की की है। है। के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के में के के में के के की है। के के

पात्र

- रिचर्ड एटनबरो
- जेफ गोल्डब्लम
- सैमुअल एल जैक्सन
- सैम नील
- लौरा डर्न
- एरियाना रिचर्ड्स
- वेन नाइट
- बीडी वोंग
- जोसेफ माजेलो
- मार्टिन फेरेरो
- बॉब पेक
- कैमरन थोर
- गेराल्ड आर मोलेन
- डीन कुंडी
- मिगुएल सैंडोवाल
- रिचर्ड केली

जीते हुए पुरस्कार

- सर्वश्रेष्ठ ध्वनि संपादन के लिए अकादमी पुरस्कार
- सर्वश्रेष्ठ ध्वनि मिश्रण के लिए अकादमी पुरस्कार
- सर्वश्रेष्ठ दृश्य प्रभाव के लिए अकादमी पुरस्कार
- राष्ट्रीय फिल्म रजिस्ट्री

पुरस्कार नामांकन

- सर्वश्रेष्ठ ध्वनि संपादन के लिए अकादमी पुरस्कार
- सर्वश्रेष्ठ ध्वनि मिश्रण के लिए अकादमी पुरस्कार
- सर्वश्रेष्ठ दृश्य प्रभाव के लिए अकादमी पुरस्कार

इन्फोबॉक्स

शैली

विज्ञान कथा फिल्म

साहसिक फिल्म

एक्शन फिल्म

उपन्यासों पर आधारित फिल्म

निर्देशक

स्टीवन स्पीलबर्ग

निर्माता

कैथलीन कैनेडी

गेराल्ड आर मोलेन

स्टीवन स्पीलबर्ग

कास्ट मेंबर

रिचर्ड एटनबरो

जेफ गोल्डब्लम

सैमुअल एल जैक्सन

सैम नील

लौरा डर्न

एरियाना रिचर्ड्स

वेन नाइट

द्वारा वितरित

यूनिवर्सल स्टूडियोज

उत्पादन कंपनी

एम्ब्लिन एंटरटेनमेंट

प्रकाशन तिथि

Figure 2: Final Wikipage generated for Jurassic Park(Template Sentence + NLG Output

Thus, we have come to an end of the report. Eventhough, we have faced many difficulties, we have still managed to produce enough content in a Wikipedia Page to call it a page unlike the LSJbot which could only generate 2-3 sentences. We would like to thank Prof. Vasudeva Verma for his continued support and our mentors, Nikhil Pattisapu and Tushar Abhishek for their feedback and help.

7 References

[1] Sam Wiseman, Stuart M. Shieber, Alexander M. Rush.
Challenges in Data-to-Document Generation.

[2] Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi.
Text Generation from Knowledge Graphs with Graph Transformers

[3] Ratish Puduppully, Li Dong, Mirella Lapata
Data-to-Text Generation with Content Selection and Planning

[4] Diego Marcheggiani, Laura Perez-Beltrachini
Deep Graph Convolutional Encoders for Structured Data to Text Generation

[5] Remi Lebret, David Grangier, Michael Auli
Neural Text Generation from Structured Data with Application to the Biography Domain