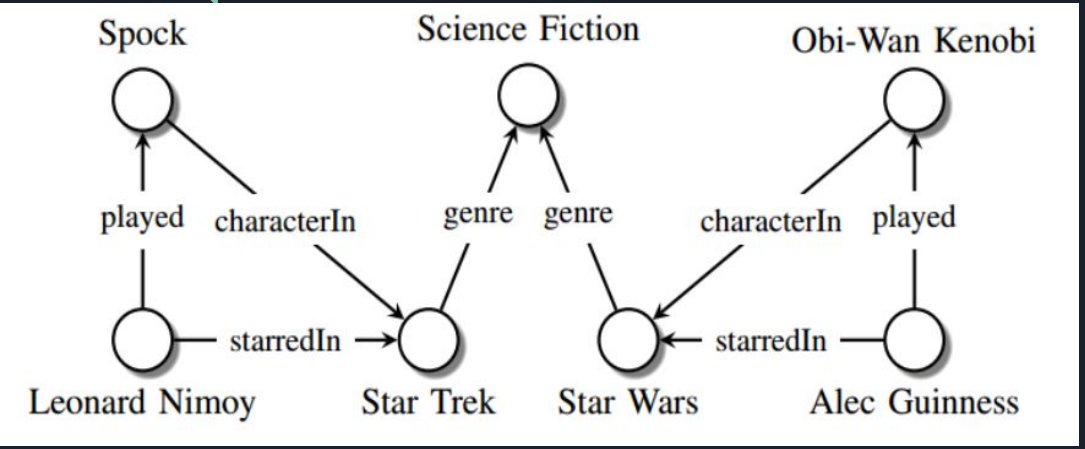
Text Generation from Knowledge Graphs like Wikidata

Aditya Agarwal(20161104)
Aniruddha Deshpande(20161058)

Knowledge Graphs

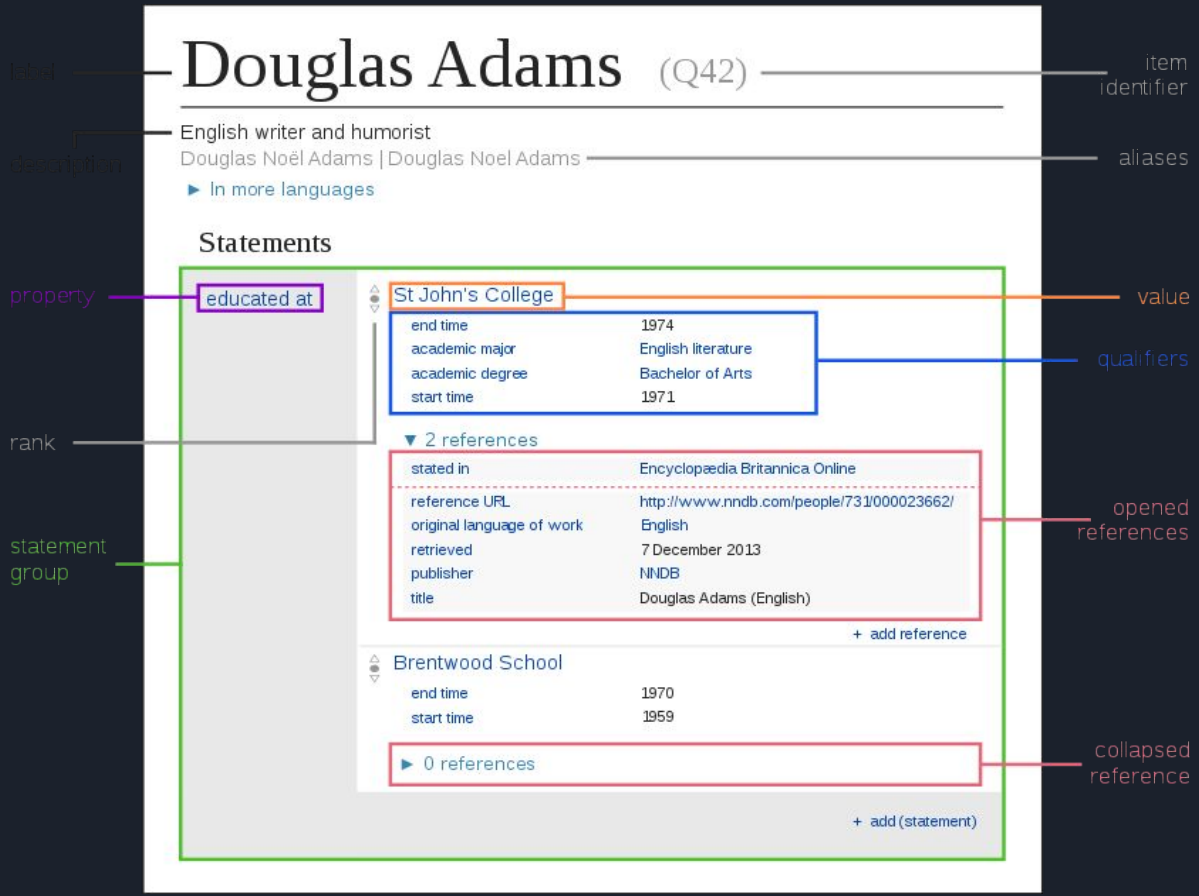


A knowledge graph (KG) represents a collection of interlinked descriptions of entities – real-world objects, events, situations or abstract concepts – where:

1. Descriptions have a formal structure that allows both people and computers to process them in an efficient and unambiguous manner.
2. Entity descriptions contribute to one another, forming a network, where each entity represents part of the description of the entities, related to it.

What is Wikidata?

- 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.



SPARQL Query Example:

```
SELECT ?human ?label
WHERE
{
 ?human wdt:P31(instance of)
wd:Q15632617(fictional
being);
        rdfs:label ?label.
 FILTER(LANG(?label) = "en").
 FILTER(STRSTARTS(?label,
"Mr. ")).
}
```

Introduction / Aim

- This project of Automatic text generation from Knowledge graphs aimed 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 came 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, coherence and discourse-relatedness is an open challenge.

- Our aim was to generate these Wikipedia pages as human-like as possible. Unlike the already existing Lsjbot, our project also aimed at generating larger documents (At least 300 Words) including all the relevant information. Lsjbot failed in doing so as it generated several documents spanning only upto one or two lines.

Stages for Our Conversion

There are mainly five stages we followed:

- Studied Wikidata and learnt about it's structure and chose a domain to work in so that we can create our Hindi Wikipedia Pages in that language and retrieving all the available data from Wikidata in that domain so that we can train on that data and then test on new data.
- Converted the data downloaded into Human Readable Format so that after training, we get it in the format that we require.
- To get it into the format required for training, we needed the Wikipedia Pages Content from this dump and the labels that we would convert in the step below as the two inputs to the NLG.
- Created a NLG code for training purposes. Also, created template sentences.
- Finally, testing and improving our final page created.

Stage 1(Wikidata Download)

- To start our conversion, we initially studied Wikidata to its bare details, identifying its structure, how data is stored etc.
- Next, we needed a domain to work in, so we were initially confused between food and Monuments but finally chose “**Monuments”** as our domain.
- Our decision of choosing Monuments, was based on the results that we obtained from looking at the results of the SPARQL queries.
- Dumping all available English Monument Data by scraping Wikidata into a JSON file, we found that we had around 11524 monument data. We then checked each page with the Hindi Wikipedia and found out that only 284 out of these had Hindi Wikipedia Pages.

Stage 2(Converting to Human Readable Format)

- We then converted the given dump(with id's) into Human Readable Format i.e. changed the identifiers(ID's) present into English Text(Labels) so that it becomes easier for us to train the NLG.
- To achieve this task, we had to write extensive code to reach the last layer of the values we needed because wikidata stores data in the form of dictionaries with each property having a sub dictionary and this goes on and on.
- We found a library called QWikidata that could get the entity dictionary from the ID and give us the label

Major Hurdle

- To code the converted JSON dump required a lot of effort and took away most of my 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. [Solution Discussed further]
- An example of the number of iterations is given above.
- Also during the entire execution, there existed some properties like P727, P1619 which had no wikidata and hence gave an error.

Stage 3(Converting into Suitable Format)

- This step involved simplifying the code that we earlier wrote into only content that we required like for example in the previous slide, we don't consider the references and hence save 2-3 iterations.
- To get it into NLG training format, we needed the labels in English initially, as we thought 284 articles were never enough for giving us good accuracy and then we found out that out of 11000 articles in English, only 2000 of those had Wikipedia Pages so we needed to find all labels of those.
- Now a huge problem was that there were some labels that did not have a wikidata page so our module gave an error and we had to write separate code to deal with those cases.
- After doing that, to get our code into NLG format, we needed a box format type execution which can be seen in the next slide.

```
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
```

- So as, we can see above, this was the required format for our labels and we had the labels in a different format altogether. Thus we converted each property, value of each monument into this format.
- We then took this box format and the wikipedia content that we had already converted and put that for training.
- Due to unforeseen circumstances though, this domain gave us some extremely bad accuracy and was not fit to be used for training. Some of the reasons were, the data lacked coherence and was very less to begin with

Stage 4 - (NLG)

Table:

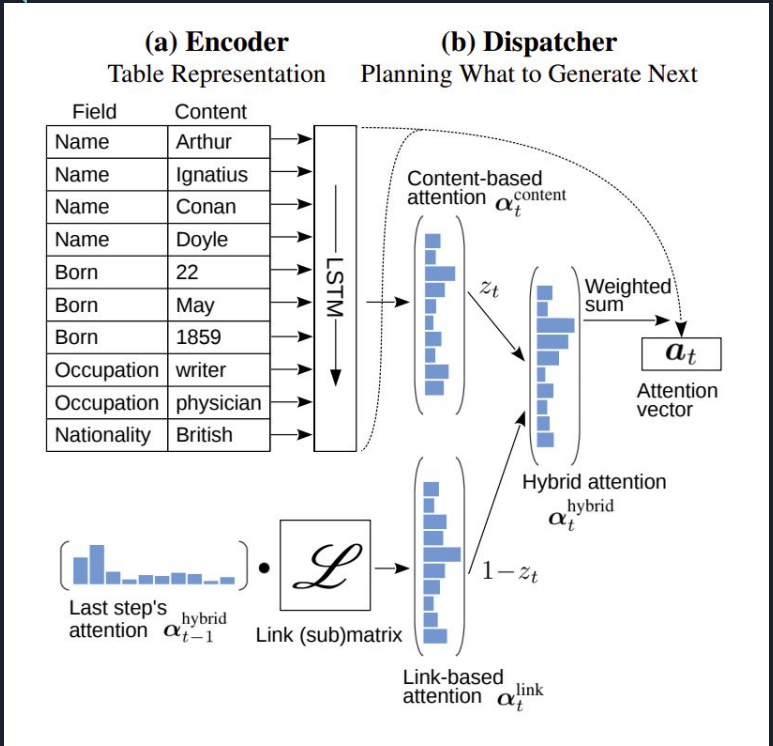
ID	Field	Content
1	Name	*Arthur Ignatius Conan Doyle*
2	Born	*22 May 1859 Edinburgh, Scotland*
3	Died	*7 July 1930 (aged 71) Crowborough, England*
4	Occupation	*Author, writer, physician*
5	Nationality	*British*
6	Alma mater	*University of Edinburgh Medical School*
7	Genre	*Detective fiction fantasy*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

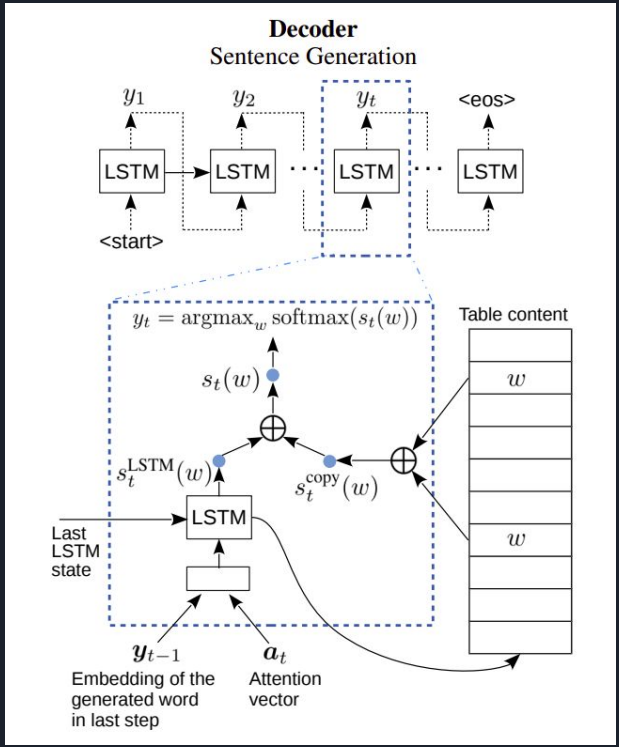
- We worked on an implementing an NLG using structured data.
- The implementation is based on the paper: "**Order-Planning Neural Text Generation From Structured Data**"
- Focus on "**Order of Contents**"
- Trained over the WikiBio Dataset.

Order-Planning Neural Text Generation From Structured Data



- Focuses on Order of content during text generation.
- Components of the Network:
  - Encoder
  - Dispatcher:
    - Content-Based attention
    - Link-Based attention
    - Hybrid attention

Order-Planning Neural Text Generation From Structured Data Continued



- Decoder variants:
  - With Copy Mechanism
  - Without Copy Mechanism
- Results achieved on the WikiBio Dataset:

Component	BLEU	ROUGE	NIST
Content att.	41.38	34.65	8.57
Link att.	38.24	32.75	8.36
Hybrid att.	43.01	36.91	8.75
Copy+Content att.	41.89	34.93	8.63
Copy+Link att.	39.08	33.47	8.42
Copy+Hybrid att.	**43.91**	**37.15**	**8.85**

Major Hurdle

- **Lack of Data** was our major problem. An NLG requires a lot of data to train but due to number of Hindi 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.
- The outputs may turn out to be extremely poor as some Wiki Pages are extremely small without much content.

Domain Change

Considering how monuments did not work for us, we now needed a domain that already had lot of data and which had uniformity which we found in Films.

We had over 2 lakh Film Data in English and around 5000 Films in Hindi Data. These were excellent parameters that we needed for our training.

We repeated all the steps as we did for the monuments except for some steps. For films, since we already had a lot of data for Hindi, we decided to train in Hindi but a major problem was that the labels were in English.

So we used the Google translate API to automatically translate those English labels into Hindi so the entire wikidata information can be stored in Hindi.

Problems in Translate API's

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, at the end to get the image in the previous slide, we had to manually translate over 60000 unique labels which took us a lot of time.

Thus after manually translating all the labels, we converted all the English Labels above into Hindi Labels except Numbers and Links.

```
{'director': ['Orson Welles'],
 'IMDb ID': ['tt0033467'],
 'Commons category': ['Citizen Kane (1941 film)'],
 'screenwriter': ['Orson Welles', 'Herman J. Mankiewicz', 'John Houseman'],
 'cast member': ['Orson Welles',
  'Joseph Cotten',
  'Ruth Warrick',
  'Everett Sloane',
  'William Alland',
  'Paul Stewart',
  'George Coulouris',
  'Dorothy Comingore',
  'Agnes Moorehead',
  'Ray Collins',
  'Fortunio Bonanova',
  'Erskine Sanford',
  'Gus Schilling',
  'Alan Ladd',
  'Charles Bennett',
  'Gino Corrado',
  'Harry Shannon',
  'Sonny Bupp',
  'Walter Sande',
  'Philip Van Zandt',
  'Roland Winters'],
 'director of photography': ['Gregg Toland'],
 'production company': ['RKO Pictures'],
```

```
{'निदेशक': ['ऑर्सन वेल्स'],
 'आईएमडीबी आईडी': ['टीटी0033467'],
 'कॉमन्स श्रेणी': ['नागरिक केन (1941 फिल्म)'],
 'पटकथा लेखक': ['ऑर्सन वेल्स', 'हरमन जे Mankiewicz', 'जॉन हाउसमैन'],
 'कास्ट मेंबर': ['ऑर्सन वेल्स',
  'जोसेफ कॉटन',
  'रूथ वारिक',
  'एवरेट स्लोएन',
  'विलियम ऑलएंड',
  'पॉल स्टीवर्ट',
  'जॉर्ज कूलूरिस',
  'डोरोथी Comingore',
  'एग्नेस मूरहेड',
  'रे कोलिंस',
  'फोर्टूनियो बोनानोवा',
  'एर्स्कीन सैनफोर्ड',
  'गस शिलिंग',
  'एलन लाड',
  'चार्ल्स बेनेट',
  'गिनो कोराडो',
  'हैरी शैनन',
  'सन्नी बुप',
  'वाल्टर सैंडे',
  'फिलिप वान Zandt',
  'रोलाण्ड विंटर्स'],
 'फोटोग्राफी के निदेशक': ['ग्रेग टॉलैंड'],
 'उत्पादन कंपनी': ['RKO पिक्चर्स'],
```

Conversion From English to Hindi

To finally get the data into the format, we had already translated the given labels from English to Hindi, we also needed the WIkidata information of each of the WIkipedia Pages we found.

The number of Wikipedia pages were found to be 4100 out of 5000. We needed 3 files in total, One containing all information about Wikipedia Pages, one containing all Wikidata about the 5000 films and lastly one file containing the Wikidata of each of those films that had a Wikipedia Page.

Thus, finally we were ready to use all these 3 files to create the box format that is required for NLG training and we started training the NLG as mentioned in Step 4.

Stage - 5 (Final Output)

DEMO

Conclusion

- We plan to make the NLG better by either training it further, bringing in some template based sentences and if time permits try out the GCN method.
- The GCN method is a Graph Convolution Network that is used to generate text from Knowledge Graphs.
- Our demo showed great potential for an output based on the difficulties we faced.
- We would like to thank Prof. Vasudeva Verma for his continued support and the mentors, Nikhil Pattisapu and Tushar Abhishek for all their help.

THANK YOU