EDGE LABELLING IN NARRATIVE KNOWLEDGE GRAPHS

Vani Kanjirangat & Alessandro Antonucci
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), USI-SUPSI, Lugano, Switzerland

INTRODUCTION

- **Knowledge Graph (KG)** creation is one of the challenging tasks, still having a wide scope of exploration.
- The automatic unsupervised creation of KG makes the process even more difficult.
- In unsupervised set up, the main steps include identifying the nodes (main entities) of KG and the relations between the nodes identified as the edges and identifying the edge labels.
- **Labelling of edges** is the main focus of proposed work.

INTRODUCTION

- The focus of proposed work is on **literary text understanding**, where the data include novels, short stories etc.
- The main problem with this domain is **lack of annotated dataset** and hence the set up is completely unsupervised.
- Another major issue is the **evaluation**.

METHODOLOGY

- The entities/ nodes here are the characters/actors in the novel/story.
- The edges represent the relations between the characters.
- The main task is then identifying the labels that could possibly describe multiple relations, which is then represented as a triplet.

➔ The challenge is how to assign a single edge label to clusters of relations.

➔ We use supersense and hypernym relations to label the edges represented using BERT embeddings.

➔ We discuss preliminary results to understand the potential for further research.

METHODS: STEP 1

Entity extraction

- Entities are the characters in the given input novel or short story, which exhibit various characteristics and relations.
- Stanford Named Entity Recognition Tagger together with a character de-aliasing is used.
  - Used DBSCAN clustering algorithm for unifying the character names (Ron and Ronald Weasley)
  - Finally , each character entity is represented by a unique identifier.

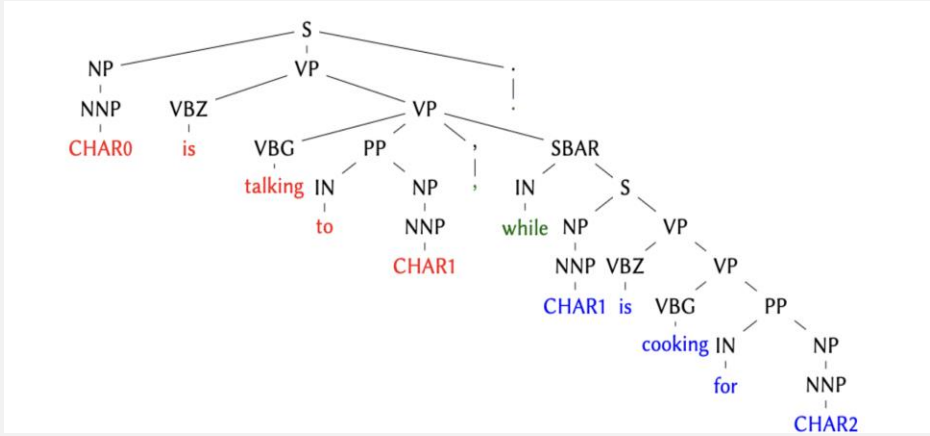
METHOD: STEP 2

Relational Sentence extraction

- Extracting the sentences that convey some relations between the entities
- We extract all the sentences with two character names excluding self-relations
- To do sentence simplification (splitting compound sentences), we used the constitueny parsing
  - Our approach traverses using DFS search and extracts each phrase (S) containing at least one noun phrase (NP) and one verb phrase (VP) starting from the bottom of the tree.

METHOD:STEP2

S= "CHAR0 is talking to CHAR1, while CHAR1 is cooking for CHAR2"



*Extract sentences: [*CHAR0 is talking to CHAR1, CHAR1 is cooking for CHAR2]

They are referred as relational sentences.

METHOD: STEP 3

Sentence Representations and Verb Extractions

- For each sentence, we encode them using SentenceBERT (SBERT)
- We then identify the VERB in these sentences and extract their corresponding embedding.
- Once we have the embeddings of all verbs, we cluster them.
- We use a two-level clustering approach.

CLUSTERING LEVEL 1

- In the first level, we group the extracted verbs based on their supersense category.
- Supersense (SS) is a terminology from WordNet, where the words are grouped into sets of synonyms called *synsets*.
- Each synset is associated with one of the 45 broader semantic categories/SSs, out of which we have 26 nouns, 16 verbs, 3 adjectives, and 1 adverb.
- It implies a coarse-grained word sense grouping approach.
- Since a word can have many senses, SS tagging and disambiguations is a challenging task.

CLUSTERING LEVEL 1

- As we consider the verbs in the sentence to describe the final triplet, the focus is on verb SS only.
- 16 verb supersense categories are considered: *{body, change, cognition, communication, competition, consumption, contact, creation, emotion, motion, perception, possession, social, stative, weather}*

Algorithm 1: Verb Clustering (Level 1)

Input: Extracted Verbs [V_1,V_2,..V_n], Supersense Categories (SC)
Output: Supersense-based Verb Clusters

```
Find embeddding of [V1,V2,..Vn];
if Verb in single SC then Assign it to that SC;
else if Verb in multiple SCs then
    for SC do
        Remove the verb from SC;
        Compute average embedding of SC with the remaining verbs;
if Verb not in any SC then Compute the average embedding of SCs;
Compute distance between the verb embedding and the average embeddings of SCs;
Assign the verb to the SC at minimum distance;
```

CLUSTERING LEVEL I

- The extracted verbs need to clsutered or tagged to the supersense categories
- If the input verb belongs to a unique SS category, we directly assign it to the SS.
- If the verb belongs to multiple SS, we compute the average embeddings for all the verbs belonging to each SS category by removing the specific input verb from the list of verbs.
- If the verb belongs to none of the SS category, we compute the average embeddings of all SS category (store them for future computations)
- Then , compute the cosine distances between input verb embeddings and the respective average embeddings
- Assign the verb to the category at minimum distance

CLUSTERING LEVEL 2

- The input to Level 2 is the SS-based cluster of verbs.
- For each cluster, we compute the *lowest-common-hypernyms* (LCHs) for every verb pair in the cluster.
  - LCH computes the lowest common ancestor node between the given synsets in the hierarchy .
- Here the problem is that a verb pair can have multiple LCHs.
- To get the most suitable LCH
  - We sort them based on their frequency of occurrences.
  - And associate the verb to the most common LCH

CLUSTERING LEVEL 2

- For each verb pair, we need to find the LCH of all the synset pairs associated with the verb pair.
- We create a dict() such that for each LCH and add the verb pair associated with the LCH

➔ For instance for LCH *Synset('give.v.03')*, we have

```
Synset('give.v.03'): [('fill', 'spend'), ('fill', 'save'),
('fill', 'give'), ('fill', 'buy'), ('fill', 'shoot'),
('spend', 'give'), ('save', 'give'), ('give', 'buy'), ('give',
'shoot')]
```

- But these verb pairs can be also part of other synset groups

```
Synset('transfer.v.05'): [('fill', 'pass'), ('fill', 'give'),
('spend', 'pass'), ('spend', 'give'), ('save', 'pass'), ('save',
'give'), ('pass', 'give'), ('pass', 'buy'), ('pass', 'shoot'),
('give', 'buy'), ('give', 'shoot')]
```

CLUSTERING LEVEL 2

- To get a unique LCH (associate a verb pair to a single LCH group), we use a simple heuristic approach
  - Sort the LCH's based on their frequency, for instance the LCH Synset('give.v.03') appears 47 times (LCH of synset pairs within different verb pairs).
  - The Synset('transfer.v.05') occurs 38 times.
  - We flatten the verb pairs as a verb list for each synset and iteratively remove the verbs if it is already present in a most frequent LCH.
  - Finally we have for `Synset('give.v.03'): ['spend', 'buy']`

CLUSTERING LEVEL 2

- Finally, we derive clusters

```
{Synset('move.v.02'): ['save', 'call', 'pass', 'give',
'take', 'hang', 'roll', 'shoot'], Synset('change.v.02'):
['fill'], Synset('act.v.01'): ['share'], Synset('give.v.03'):
['spend', 'buy'], Synset('get.v.01'): ['borrow']}
```

- The final edge label is the associated LCH.

CLUSTERING LEVEL 2

Algorithm 2: Verb Clustering (Level 2)

Input: Supersense-based Verb Clusters
Output: Triplet (*C*1, *r*,*C*2)

```
for each supersense-based verb cluster do
    Take all the verb pairs
for each verb pair do
    Compute the lowest common hypernym (LCH) and store them all;
    Sort the LCHs based on their frequency;
for each verb do
    Associate it to the most common LCH;
    if no LCH associated to a verb then Consider as outlier;
    Associate the relation label with the corresponding LCH;
    Generate the triplets;
```

EXPERIMENTS

- We did the preliminary experiments on the six books in Harry Potter Series by J.K. Rowling (885‘943 words)
- The statistics at different steps of relational sentence detection:

Type of Sentences	# Before/After Co-Referencing
Identified sentences	6394/6394
With two chars	566/618
Asymmetric sentences	511/564
Two different chars	470/516
Not included sentences	470/516
Not “... said charX...”	387/433
Verb between chars	331/380

EXPERIMENTS

- Results with Clustering Level-1 and 2

Verb Category	Verbs
stative	{shake,lose,study,relax,favor}
communication	{speak,raise,bully,cheer,warn,mutter}
consume	{growl,scramble,eat}
motion	{move,walk,slip,look}
emotion	{fuss,cast,recognize,scare}
possession	{hand,find,clap,save,borrow,award,swap}
body	{smile,laugh,grin,blink,spit}
perception	{fill,whip,fight,insist,glance,throw,break}
cognition	{snore,hear,feel,help,share,gasp,linger,dance}
social	{celebrate,dare,punish}

Verbs	Triplets
play, act	(Harry,play,Slytherin)
	(Harry,act,Snape)
complain, mutter	(Harry,mutter,Snape)
	(Ron,mutter,Harry)
block,fight	(Marcus,block,Harry)
	(Granger,fight,Snape)
say,repeat	(Quirrell,say,Snape)
	(Ron,repeat,Hagrid)

CONCLUSION & FUTURE WORK

- The study was just a preliminary attempt for unsupervised triplet creations, specifically edge labelling.
- We have experimented it only in literary text domain, on a novel series.
- As future work, we consider experimenting with better approaches such as representations uisng Sense-BERT.

THANK YOU!