Quora question similarity

Ronak, Snehil

Problem

Given a pair of questions, we have to tell if these questions are semantically similar or not.

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0

Data set

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0
5	5	11	12	Astrology: I am a Capricorn Sun Cap moon and c...	I'm a triple Capricorn (Sun, Moon and ascendan...	1
6	6	13	14	Should I buy tiago?	What keeps childern active and far from phone ...	0
7	7	15	16	How can I be a good geologist?	What should I do to be a great geologist?	1
8	8	17	18	When do you use シ instead of し?	When do you use "&" instead of "and"?	0
9	9	19	20	Motorola (company): Can I hack my Charter Moto...	How do I hack Motorola DCX3400 for free internet?	0

Dataset

- Number of training data: 404290
- Number of duplicate question pairs in total data : 149263
- percentage of duplicate question pairs in total data : 36.919800 %
- Total number of questions: 537933
- Number of questions appearing multiple times: 111780

Approach

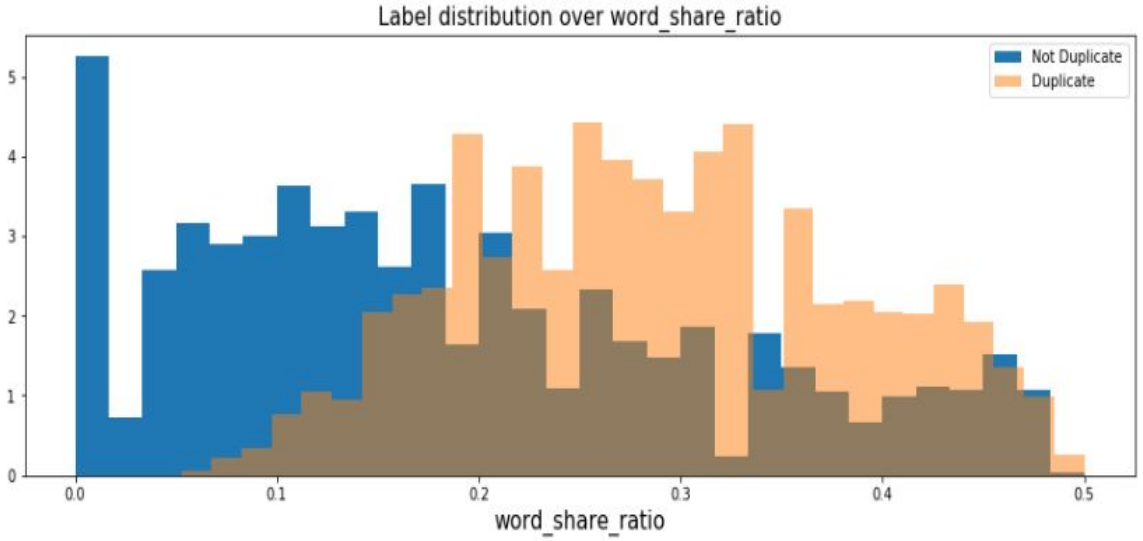
Common words

Supervised learning

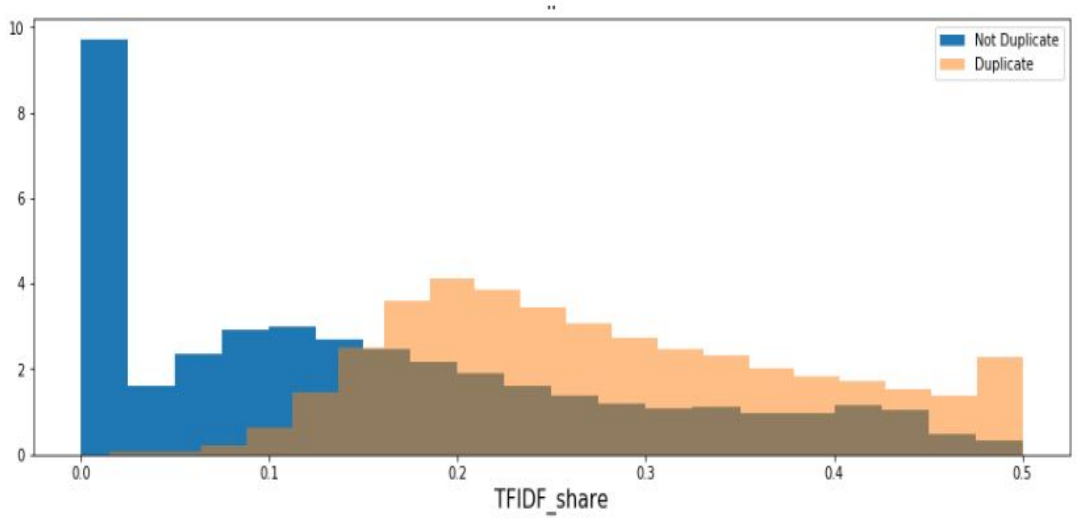
Features extracted

- Number of words
- Character length
- Ratio of common words
- WMD distance
- Tf-Idf weighted word share

Common word share ratio



Tf-Idf weighted word share



Bag Of Words Model

- Convert the raw data into tf-idf features
- Ngrams range - 1, 2, 3, 4
- Minimum Document Frequency - 0, 10, 100, 200, 400, 600
- Classifiers used - Logistic Regression, XGBoost

LSTM with Attention

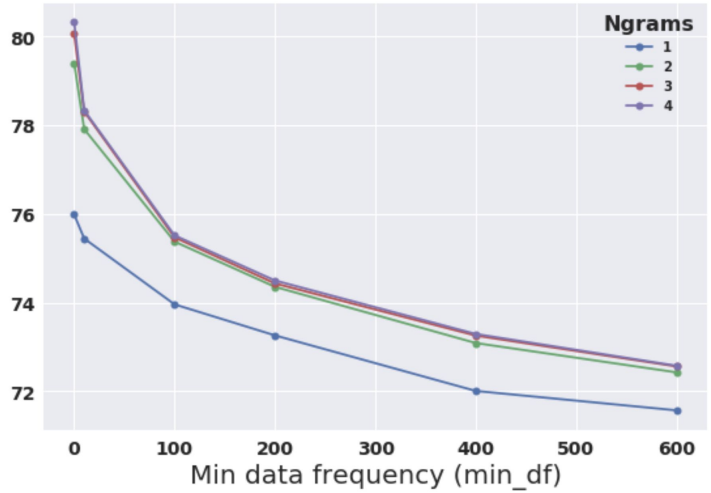
- 2 layers of biLSTM
- Attention layer
- GLOVE word embedding
- Optimizer - Adam
- Loss function - Binary cross entropy

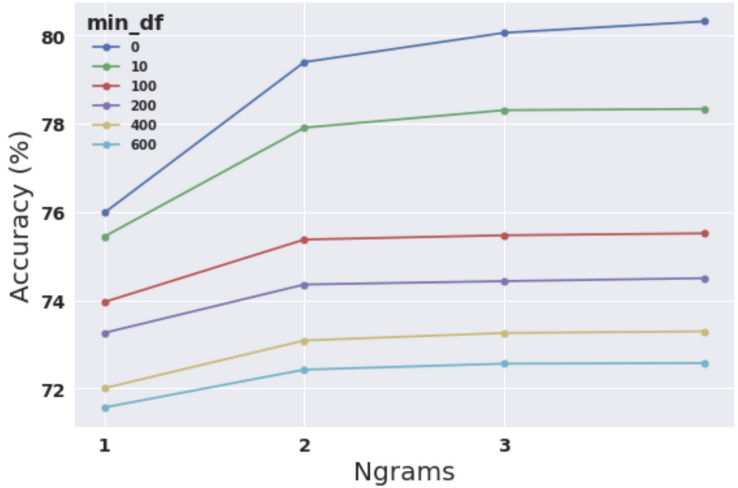
Results

- Ran various supervised learning algorithms on the handcrafted Features.
- SVM does worst, data is not linearly separable.
- XGboost does well.

Model	Result
Random Forest (baseline)	73.9%
Logistic Regression	66.6 %
Decision Tree	69.3%
SVM	37.8%
KNN	71.7%
XGboost	81.8%
Bag-of-words Logistic Regression	80.32%
Bag-of-words XGBoost	78.2%
LSTM with attention	79.58%

Results


Ngrams
1
2
3
4
80
78
76
74
72
0
100
200
300
400
500
600
Min data frequency (min_df)


min_df
0
10
100
200
400
600
Accuracy (%)
80
78
76
74
72
1
2
3
Ngrams

Conclusion

- All our proposed models performed much better than the base-line model used by Quora.
- Unexpectedly, traditional machine learning are performing better than the deep learning ones.
- This can be attributed to fewer number of iterations run by us.

Deep Clinic

By Ashok Govinda Gowda
Saagar Minocha

Problem Statement

Clinical Concept Extraction involves extraction of clinical concepts in order to transform unstructured data from clinical notes to structured information with automatic annotations. Clinical notes, in particular, use complex natural language constructs which are hard to automatically process and understand.

For example:

Ashok is suffering from fever and cough. He took a ECG test and was prescribed tylenol on November 26, 2018.

The concept annotations are [‘problem’, ‘treatment’,’test’]

In this case, The concepts extracted would be fever and cough for problems, ECG test for test, tylenol for treatment.

Challenges

- Ambiguity of medical terms and concepts,
- Use of non-standard acronyms
- Out of vocabulary words
- Lack of good annotated corpus
- Complexity of sentence structures
  - conditional constructs (denoting future events)
  - negated constructs (denoting absence of concepts)
  - uncertain constructs (denoting uncertain events)
  - historical constructs (denoting patient's history)
  - familial constructs (denoting patient's genetical history)

Proposed Solution Overview

- Data Preprocessing
- BIO-Tagging
- Baseline Approach (using CRF)
- Parameter Tuning
- Deep Learning Approach (using CNN-BiLSTM)
- Experimenting with Word Embeddings

DataSet

Informatics for Integrating Biology and the Bedside (i2b2) 2010 dataset
https://www.i2b2.org/NLP/DataSets/Main.php

i2b2 2010/VA	Training	Test
Summaries	170	256
Sentences	16,414	27,613
Problems (Annotated Entities)	7,073	12,592
Treatments (Annotated Entities)	4,844	9,344
Test (Annotated Entities)	4,608	9,225

Data Formats

Line # Input Text

25 The patient is a 63-year-old female with a three-year history of bilateral hand numbness and occasional weakness .

26 Within the past year , these symptoms have progressively gotten worse , to encompass also her feet .

27 She had a workup by her neurologist and an MRI revealed a C5-6 disc herniation with cord compression and a T2 signal change at that level .

Concepts

c="bilateral hand numbness" 25:11 25:13||t="problem"
c="occasional weakness" 25:15 25:16||t="problem"
c="these symptoms" 26:5 26:6||t="problem"
c="a workup" 27:2 27:3||t="test"
c="an mri" 27:8 27:9||t="test"
c="a c5-6 disc herniation" 27:11 27:14||t="problem"
c="cord compression" 27:16 27:17||t="problem"
c="a t2 signal change" 27:19 27:22||t="problem"

Assertions

c="bilateral hand numbness" 25:11 25:13||t="problem"||a="present"
c="occasional weakness" 25:15 25:16||t="problem"||a="present"
c="these symptoms" 26:5 26:6||t="problem"||a="present"
c="a c5-6 disc herniation" 27:11 27:14||t="problem"||a="present"
c="cord compression" 27:16 27:17||t="problem"||a="present"
c="a t2 signal change" 27:19 27:22||t="problem"||a="present"

Relations

c="an mri" 27:8 27:9||r="TeRP"||c="a c5-6 disc herniation" 27:11 27:14
c="an mri" 27:8 27:9||r="TeRP"||c="cord compression" 27:16 27:17
c="an mri" 27:8 27:9||r="TeRP"||c="a t2 signal change" 27:19 27:22
c="a c5-6 disc herniation" 27:11 27:14||r="PIP"||c="cord compression" 27:16 27:17

BIO tagging

BIO tagging is a format for tagging tokens in a chunking task . The B- prefix before a tag indicates that the tag is the beginning of a chunk, and an I- prefix before a tag indicates that the tag is inside a chunk. The B- tag is used only when a tag is followed by a tag of the same type without O tokens between them. An O tag indicates that a token belongs to no chunk.

For Example:

Ashok is suffering from fever cough

('Ashok, 'O'), ('is', 'O'), ('sufferning', 'O'), ('from', 'O'), ('fever', 'B-problem'), ('cough, 'I-problem'),

Conditional Random Fields
Baseline Approach

As a Baseline Approach, Conditional Random fields was used.

Conditional Random fields are utilized in places where we have to take sequential information into account. For example: In the previous sentence we had B-problem which would be followed by any number of I-problems. We are taking this into account with Conditional Random fields.

For this we used the sklearn-crfsuite library to utilize conditional random fields.

Feature Creation

```
{'LOWERCASE': 'of',
 'DIGIT': False,
 'LAST_TWO_LETTERS': 'of',
 'UPPERCASE': False,
 'LAST_THREE_LETTERS': 'of',
 'TITLE': False,
 'BIAS': 1.0,
 'LENGTH': 2,
 'STOPWORD': True,
 'PREVIOUS_WORD_LOWER CASE': 'end',
 'PREVIOUS_WORD_TITLE': True,
 'PREVIOUS_WORD_UPPERCASE': False,
 'NEXT_WORD_LOWER_CASE': 'report',
 'NEXT_WORD_TITLE': True,
 'NEXT_WORD_UPPER_CASE': False}
```

Training Using L-BFGS Algorithm

	precision	recall	f1-score	support
B-test	0.8696	0.7660	0.8145	9225
I-test	0.8102	0.7324	0.7693	8012
B-problem	0.8261	0.7503	0.7864	12592
I-problem	0.7372	0.7702	0.7533	17684
B-treatment	0.8435	0.7009	0.7656	9344
I-treatment	0.7396	0.6396	0.6859	7954
micro avg	0.7950	0.7350	0.7638	64811
macro avg	0.8044	0.7265	0.7625	64811
weighted avg	0.7980	0.7350	0.7639	64811

Parameter Optimization Using Grid Search

Optimization of the parameters used in the Conditional Random fields was done using Grid Search.

grid search, or a parameter sweep, which is an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set or evaluation on a held-out validation set

Here the performance metric used while performing grid search was f1-score

Analysis of Relationships

Most Common Relationships

I-problem I-problem 3.066507
I-treatment I-treatment 2.978982
I-test I-test 2.786419
B-problem I-problem 2.51442
B-test I-test 2.41495

Least Common Relationships

O I-problem -7.86171
O I-test -7.526132
O I-treatment -7.295169
B-treatment I-problem -5.269959
B-treatment I-test -4.928459

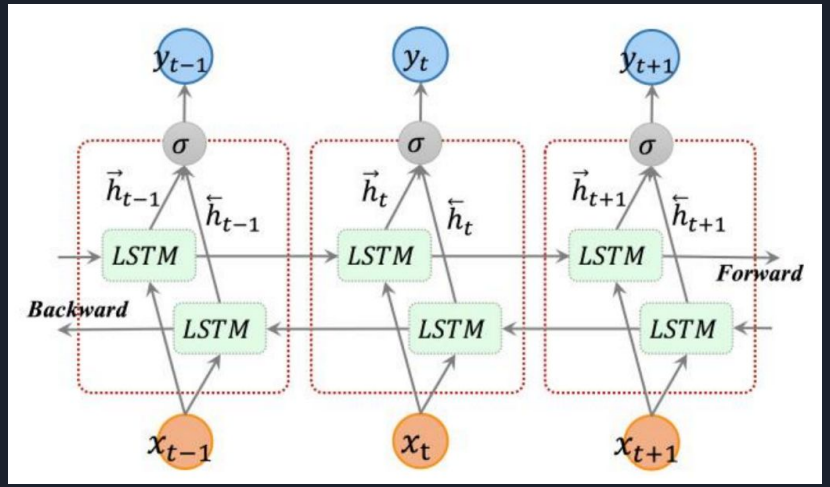
Analysis of Word Features

The Most Common Word Features were
LOWERCASE:auscultation B-test 5.774259
LOWERCASE:hypertension B-problem 5.19519
LOWERCASE:apgars B-test 5.110699
LOWERCASE:srom B-problem 4.894952
LOWERCASE:fevers B-problem 4.871558
LOWERCASE:fever B-problem 4.869986

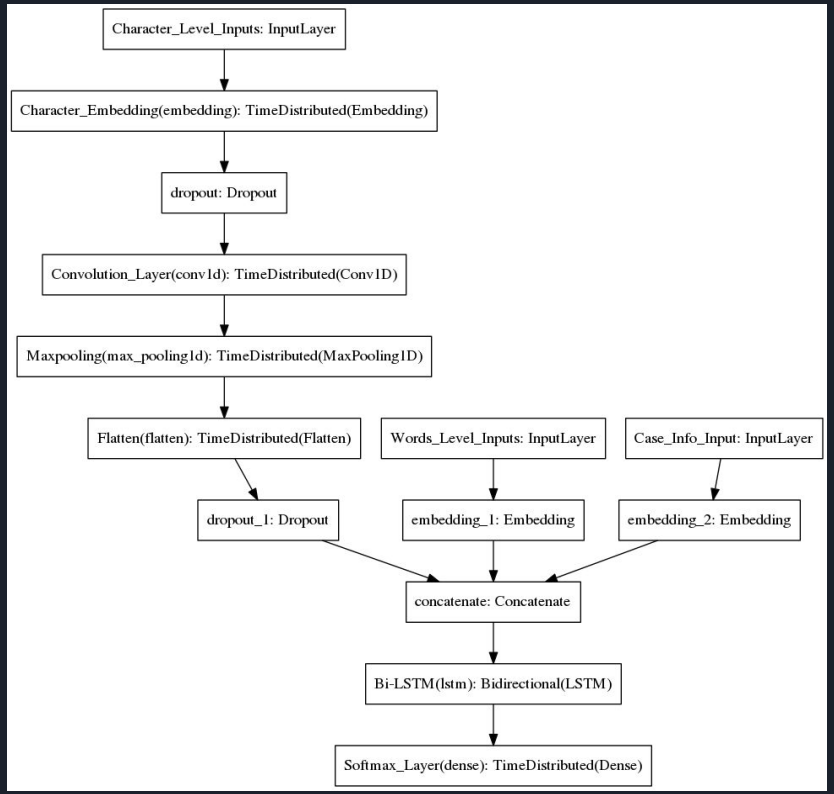
PREVIOUS_WORD_LOWER CASE:a B-problem -3.738002
PREVIOUS_WORD_LOWER CASE:the B-problem -3.506919
LOWERCASE:pain O -3.345862
LOWERCASE:sedated O -3.231368
PREVIOUS_WORD_LOWER CASE:# O -3.131723
PREVIOUS_WORD_LOWER CASE:a B-test -3.054437

Deep Learning Approach

- Character Level Embeddings (patterns identified using CNN)
- Word Level Embeddings (using GloVe or fastText vectors)
- Additional Case Information (lowercase, uppercase, numeric etc.)
- Concatenating the 3 layers and feeding to biLSTM for tagging



Model Architecture



Word Embeddings

- **Stanford's GloVe Vectors**
  glove.6B.50d - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d vectors)

  GloVe constructs a co-occurrence matrix (words X context) to count how frequently a word appears in a context in order to learn. Factorization of this big matrix is usually done to achieve a lower-dimension representation.

- **Facebook's fastText Vectors**
  wiki-news-300d-1M.vec: 1 million word vectors trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset (16B tokens).

  fastText uses n-gram characters as the smallest unit to generate better word embeddings (array of numbers with predefined dimensions) for rare words and out of vocabulary words as the n-gram character vectors are shared with other words

Results

Tuning Parameter	GloVe (60 epochs)	GloVe (30 epochs)	FastText(30 epochs)
Epochs	60	30	30
Dropout	0.5	0.5	0.5
Dropout Recurrent	0.25	0.25	0.25
LSTM State Size	200	200	200
Convolution Kernel Size	3	3	3
Optimizer	Nadam()	Nadam()	Nadam()
F1-Score	**84.88**	**84.27**	**75.16**

Conclusion

- Preprocessed raw data from an i2b2 dataset of clinical summaries
- Converting it to a BIO tagged format.
- Established a statistical baseline using Conditional Random Fields and achieved a fairly good F1-score of around 80%.
- Hyperparameter Tuning
- Composed a neural network model using a bi-LSTM and a character-level CNN achieving an F1-score of around 85% (higher than the baseline results)
- Experimented with fastText Embeddings

Thank You

Questions?

Understand action language with deep neural network

QING WAN

Instructed by Dr. Yoonsuck Choe

MOTIVATION

- Communication vs Language
- Piaget's Four stages
  - 1st: Sensorimotor Stage (birth ~ 2yrs)
  - 1st & 2nd: Object permanence & Causality
- Combine Sensorimotor control with Language

BACKGROUND

[1] introduced neural process network.

- Could track common sense attributes through neural simulation of action dynamics.
- Data set: > 120, 000 recipes (JSON).
- No codes released yet.

References:

[1] *Simulating Action Dynamics with Neural Process Networks*, Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, Yejin Choi, May, 2018, arXiv: 1711.05313.

EXAMPLE

Sausage beef ribs rigatoni (first 3 steps)

Text:

- In a large pan, sauté onions and garlic in oil until tender;
- Remove and set aside;
- Add chuck to pan, brown on all sides, then remove to onion dish;

Action:

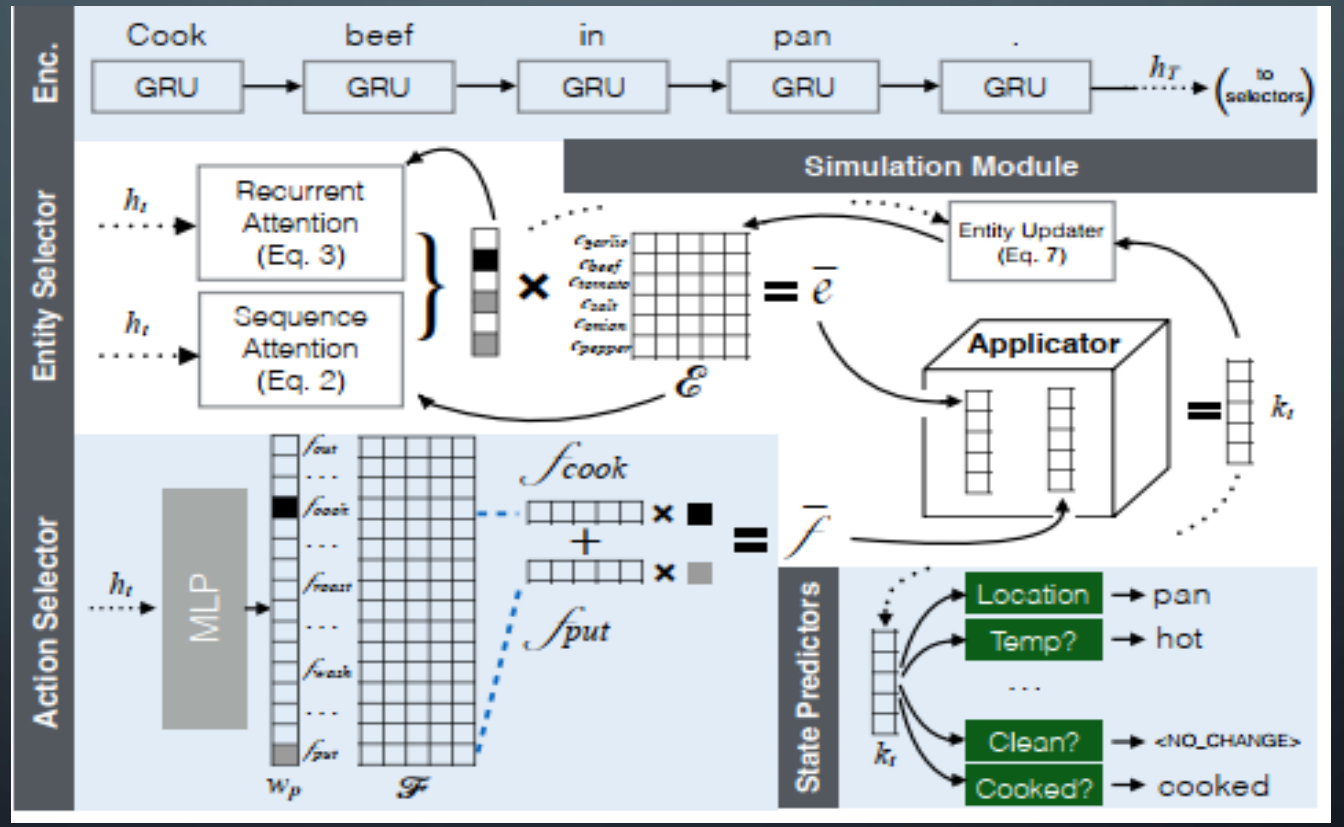
- sauté; remove; brown, add, remove.

Ingredients:

- olive oil, onion, cloves garlic; same as previous; onion, chuck cut.

ORIGINAL ARCHITECTURE

WORD2VECTOR



TRAINING AND RESULTS

- Each stage train independently
- No information about how accurate the system could achieve except for entity selector with F1 score (vague) ~55%.

F1 score defined as:

$$2 \times \frac{Precision \times Recall}{Precision + Recall} \text{ where } Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN}$$

- My trail: ingredient/action selector accuracy: ((TP+FN))/total 0.5%~1%.

A GLANCE

Good	s_{t-1}	Add tomato paste, broth, garlic, chili powder, cumin, chile peppers, and water.
	s_t	Bring to boil, then turn very low, cover and simmer until meat is tender.
	selected	meat, garlic, chili powder, tomato paste, cumin, chiles, beef broth, water
	correct	Same + [oil]
Good	s_{t-4}	Stir in oats, sugar, flour, corn syrup, milk, vanilla extract, and salt.
	s_{t-3}	Mix well.
	s_{t-2}	Drop by measuring teaspoonfuls onto cookie sheets.
	s_{t-1}	Bake 5 - 7 minutes.
	s_t	Let cool.
	selected	oats, sugar, flour, corn syrup, milk, vanilla extract, salt
	correct	oats, sugar, flour, corn syrup, milk, vanilla extract, salt
Good	s_{t-1}	In a large saucepan over low heat, melt marshmallows.
	s_t	Add sprinkles, cereal, and raisins, stir until well coated.
	selected	marshmallows, cereal, raisins
	correct	marshmallows, cereal, raisins, sprinkles
Bad	s_{t-3}	Ladle the barbecue sauce around the crust and spread.
	s_{t-2}	Add mozzarella, yellow cheddar, and monterey jack cheese.
	s_{t-1}	Next, add onion mixture and sliced chicken breast .
	s_t	Top pizza with jalapeno peppers.
	selected	jalapenos
	correct	crust, sauce, mozzarella, cheddar, monterey jack, white onion, chicken, jalapenos
Bad	s_{t-2}	Combine 1 cup flour, salt, and 1 tbsp sugar.
	s_{t-1}	Cut in butter until mixture is crumbly, then sprinkle with vinegar .
	s_t	Gather dough into a ball and press into bottom of 9 inch springform pan.
	selected	butter, vinegar
	correct	flour, salt, sugar, butter, vinegar

OTHER PROBLEMS

Four Corpora (high dimensional spaces)

- First three, text (7355), ingredient (2994), action (384) 
- Last one, states change (~380) 

ARCHITECTURE UPGRADED

WORD2VECTOR

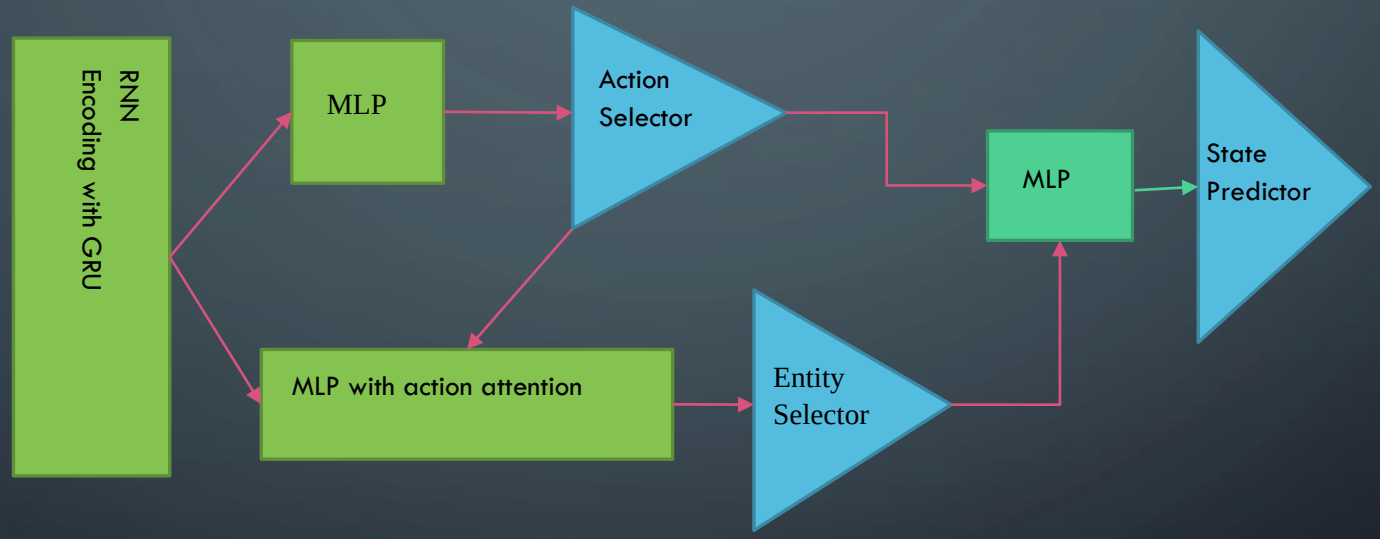


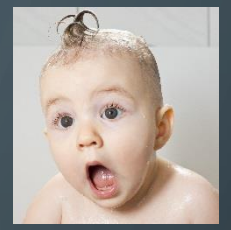
Fig 2

EXPERIMENT

- Four Corpora: by statistics, 8700, 5505, 380, 381.
- TensorFlow 1.12 with GTX 1070 GPU, > 4GB pre-allocated.
- Batch size: 50.
- 1st & 2nd stage: 250 epochs with 501 respectively; 3rd: 500 epochs with 20,000.
- Temporal consumption: 24 hrs; 48 hrs; 50 hrs (Nov 17 ~ Nov 25).

RESULTS

- Accuracies for Entity/Action selectors (incredible): 5 starts.
- Accuracy for States change (good): 4 stars. 😄



GO OVER PREVIOUS EXAMPLE

```
Action expected:  ['saute']
Action predict:  ['saute']
Entity expected:  ['onion', 'olive_oil', 'cloves_garlic']
Entity predict:  ['onion', 'olive_oil', 'cloves_garlic']
```

```
Action expected:  ['remove']
Action predict:  ['remove']
Entity expected:  ['onion', 'olive_oil', 'cloves_garlic']
Entity predict:  ['onion', 'olive_oil', 'cloves_garlic']
```

```
Action expected:  ['add', 'remove', 'brown']
Action predict:  ['add', 'remove', 'brown']
Entity expected:  ['UNK', 'onion']
Entity predict:  ['UNK', 'onion']
```

```
states predict: [('cookedness', 'cooked'), ('temperature', 'hot')]
states_expected: [('cookedness', 'cooked'), ('location', 'pan'), ('temperature', 'hot')]
```

```
states predict: [('composition', 'change')]
states_expected: [('composition', 'change')]
```

```
states predict: [('temperature', 'hot'), ('composition', 'change')]
states_expected: [('cookedness', 'cooked'), ('temperature', 'hot'), ('composition', 'change')]
```

THANK YOU!

Q&A for US Pharma Industry

by APOORV KUMAR

Index

Introduction

- Currently, in the US Pharma industry, most of the product companies have the most important promotion channel as ‘physician calls’, which are delivered by sales reps
- Now, these sales reps usually want to access certain information about the doctor they will be meeting OR their own performance in their region.
- ~30-40% of sales rep don’t use the existing systems and rely on their previous understanding and hunch.

Index

Techniques adopted

- Method 1: **Stanford SEMPRE Library** –
  - Stanford NLP group has created a Library called Sempre for the specific task of answering questions.
- Method 2: **Using pre-defined question formats** –
  - Define question formats and then match the incoming requests to one of those questions

Method 1: Sempre library

- Sempre library was adopted because –
  - **All encompassing**: By its design, it looks like one can answer a wide range of questions using a single process
  - **Scalable**: Since Sempre uses a graph database called Virtuoso and a graph querying language calling SPARQL, it should be able to instantly produce answers even with large amount of data.
  - **Less production time as range of questions grow**

Disadvantages of Sempre

- **Not for complex questions**: It is not specifically built for answering questions from tabular data, such as the one in this project.
- **Lambda DCS is not a full-fledged programming language**: The authors of this library themselves mention that the intent is not to create a complete programming language.
  - One cannot select multiple columns of data. (Ex. in the question 'show address of doctors in zip 77840', we will expect to see both names and address of doctors, but the current set of tools can only show one thing at a time (address or names)).
- **Lack of appropriate documentation**: Documentation is limited and there have been instances where some functionality was found but was not documented.

Method 2: Pre-defined question formats

- In this, several question formats are coded and then an incoming request is matched to one of those questions.
- On the implementation side, a question is converted to a set of filters, groupby columns and the columns to be selected.
- These parameters are sent to the server running in Flask in Python.
- MongoDB database is used since it provides the flexibility to incorporate the varied data in the project.

Index

- Introduction
- Techniques adopted
- Results and discussion
- Conclusion

Results and discussion

- Method 1: A sample of results from automated testing are as desired –
  - The time is in ‘ms’

	B	C	D
1	**questions**	**answer**	**time**
2	doctors in 00601	NameSMITH WOODRIGOBERTO RAMOS GONZALEZJOSE A ROMAN RAMOS	428
3	doctors	NameSMITH WOODMANUEL A TORRES PEREZRIGOBERTO RAMOS GONZALEZZELIDETH RIVERAJOSE A ROI	63
4	show doctors in 00601	NameSMITH WOODRIGOBERTO RAMOS GONZALEZJOSE A ROMAN RAMOS	368
5	show doctors in zip 00602	NameMANUEL A TORRES PEREZZELIDETH RIVERAJOSE A ROMAN RAMOS	480
6	show names of doctors in 00601	NameSMITH WOODRIGOBERTO RAMOS GONZALEZJOSE A ROMAN RAMOS	2301
7	show doctors in college station	NameRIGOBERTO RAMOS GONZALEZJOSE A ROMAN RAMOS	393
8	show sum of sales	Sum2910	68
9	show sales	Sum2910	52
10	show sales in zip 00601	Sum1810	353
11	sales for doctor smith	Sum600	294
12	trx for doctor smith in last 3 months	SumCol1250fb:en.201807200fb:en.201808150fb:en.201809	1838
13	trx for doctor jose in last month	Sum100	1617
14	tell sales for doctors in east hanover	Sum810	1527
15	tell me the address of doc smith	Col0302 BALL ST, APT #K305, EAST HANOVER, NJ 00602	466

Index

Conclusion

- Sempre toolkit looks promising but it needs more functionality to make it useful for a question-answering project from tabular data
- The pre-defined question format method looks naïve but will be able to produce results faster in the beginning. This method will become challenging as the number of questions will grow.

Question Answering with the SQuAD dataset

CSCE 638 Final Project

Question Answering with the SQuAD dataset

- Introduction
- Infersent model
- Passage (sentence) retrieval
- Qusestion Processing
- Answer Processing
- Answer Catching
- Accuracy
- Future Work

Introduction

- What is QA task ?
- Question Types:
  - Simple questions
  - Complex questions
    - Narrative
    - Opinion

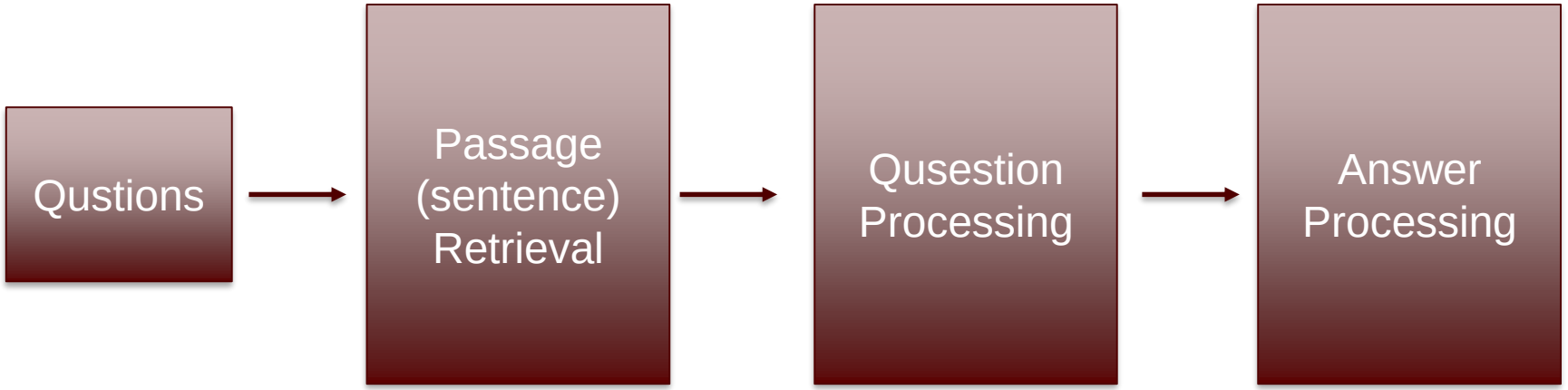
Reading Comprehension Tests

Mars Polar Lander- Where Are You?

(January 18, 2000) After more than a month of searching for a single sign from NASA's Mars Polar Lander, mission controllers have lost hope of finding it. The Mars Polar Lander was on a mission to Mars to study its atmosphere and search for water, something that could help scientists determine whether life even existed on Mars. Polar Lander was to have touched down on December 3 for a 90-day mission. It was to land near Mars' south pole. The lander was last heard from minutes before beginning its descent. The last effort to communicate with the three-legged lander ended with frustration at 8 a.m. Monday. "We didn't see anything," said Richard Cook, the spacecraft's project manager at NASA's Jest Propulsion laboratory. The failed mission to the Red Planet cost the American government more the $200 million dollars. Now, space agency scientists and engineers will try to find out what could have gone wrong. They do not want to make the same mistakes in the next mission.

- When did the mission controllers lost Hope of communication with the Lander? (*Answer:* 8AM, Monday Jan. 17)
- Who is the Polar Lander's project manager? (*Answer:* Richard Cook)
- Where on Mars was the spacecraft supposed to touch down? (*Answer:* near Mars' south pole)
- What was the mission of the Mars Polar Lander? (*Answer:* to study Mars' atmosphere and search for water)

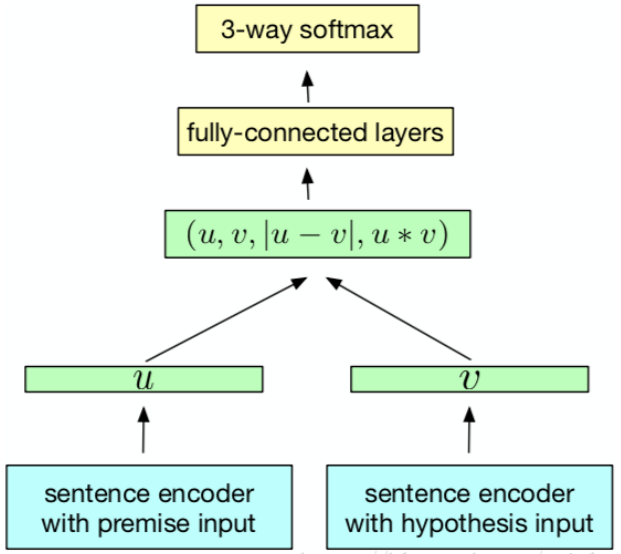
Approaches


Qustions
Passage (sentence) Retrieval
Qusestion Processing
Answer Processing

Infersent model

- InferSent is a sentence embeddings method that provides semantic representations for English sentences
- How is Infersent implemented
- Model selection : trained on

  GloVe dataset (discussed later)



Generic NLI training scheme.

Passage (sentence) retrieval on given context

- Split context by TextBlob
- Use Infersent for sentence embedding
- Evaluate similarity by distance like:
  - Cosine similarity
  - Euclidean Distance
  - Manhattan Distance
  - …

Passage (sentence) retrieval on given context

- Result:

Distance Metric	Accuracy
Cosine similarity	0.6149162861491628
Euclidean distance	0.3995433789954338
Manhattan distance	0.43262481689453125
Chebyshev distance	0.2815829528158295

Accuracy of unsupervised learning base on different distance

Passage (sentence) retrieval without context

- Get the context from Wikipedia page
- The other part is same as previous one
- Result:
  - Question form SQuAD dataset: unsatisfying
  - Custom Question: depends largely on the structure of question

Find Passage (sentence) retrieval without context

- Examples of questions from SQuAD dataset

Question 1	The Basilica of the Sacred heart at Notre Dame is beside to which structure?
Answer 1	The Basilica of the Sacred Heart in Notre Dame, Indiana, USA, is a Roman Catholic church on the campus of the University of Notre Dame, also serving as the mother church of the Congregation of Holy Cross (C.S.C.)

Passage (sentence) retrieval without context

- Examples of custom questions

Question 1	How many Nobel laureates has Oxford educated ?
Answer 1	The university is consistently cited as among the world's best.Oxford has educated many notable alumni, including 29 Nobel laureates

Model Selection : GloVe or fastText?

- Infersent V1 trained with GloVe
- Infersent V2 trained with fastText
- fastText shows poor performance

Distance Metric	Accuracy
Cosine similarity	0.2488584474885844 6
Euclidean distance	0.292998477929984 8
Manhattan distance	0.303652968036529 7
Chebyshev distance	0.2473363774733637 6

Accuracy of unsupervised learning base on different distance with Infersent trained on fastText

Model Selection : GloVe or fastText?

- Why such difference?
- GloVe treats each word as an atomic entity and generated the corresponding vector.
- fastText treats each word as composed of character n grams

Question Classification

Classes Based on OntoNotes Corpus

- 'PERSON'
- 'NORP'
- 'FACILITY'
- 'ORGANIZATION'
- 'GPE'
- 'LOCATION'
- 'PRODUCT'
- 'EVENT'
- 'WORK OF ART'
- 'LAW'
- 'LANGUAGE'
- 'DATE'
- 'TIME'
- 'PERCENT'
- 'MONEY'
- 'QUANTITY'
- 'ORDINAL'
- 'CARDINAL'

Question Classification

Rules we used:

- Queries starting with “Who” or “Whom” are taken to be of type “PERSON”;
- Queries starting with “Where”, “Whence”, or “Whither” are taken to be of type “LOCATION”;
- Queries starting with “How few”, “How great”, “How little”, “How many” or “How much” are taken to be of type “QUANTITY”;
- Queries string with “Which” or “What”, look up head noun in lexicon to determine answer type.

Question Classification

To find the head noun:

- StanfordPOSTagger
- Eliminate preposition phrases

To determine question type:

- WordNet hypernyms
- Hypernyms of hypernyms

Question Classification

Classification result:

# Determined with Rule 1, 2, 3	# Determined with Rule 4	Others
379	618	317

Preparation for Answer Catching

- Even we have the question tag, but it is not as accurate as we expect.
- We do the related-tag mapping for each question

'PERSON'→'PERSON'
'NORP'→'NORP'
'FACILITY'→'FACILITY'
'ORGANIZATION'→'ORGANIZATION'
'GPE'→{'GRE','LOCATION'}
'LOCATION'→{'GPE','LOCATION'}
'PRODUCT'→{'PRODUCT'}
'LAW'→'LAW'
'TIME'→{'DATE','TIME'}
'DATE'→{'DATE','TIME'}
'QUANTITY'→{'QUANTITY','PERCENT','MONEY'}
'PERCENT'→{'PERCENT','QUANTITY'}
'MONEY'→'QUANTITY'
'LANGUAGE'→'LANGUAGE'
'CARDINAL'→'CARDINAL'
'ORDINAL'→'ORDINAL'
'EVENT'→'EVENT'
'WORK OF ART'→'WORK OF ART'

Answer Catching

- tagged each word in target sentence using CogComp-NLP

  --CogComp-NLP provides a suite of state-of-the-art Natural Language Processing (NLP) tools that allows you to annotate plain text inputs.

- Save word which hold same tag with question into answer list

Accuracy

- Full-Match Method

--all words in expected answer shown in result

- Partial-March Method

--at least one word in expected answer shown in result

Credits

Add your credits here

Machine Reading Comprehension

Stuti Sakhi*, Yerania Hernandez†
Department of Computer Science and Engineering
Texas A & M University
*Email: stuti@tamu.edu
† Email: hernandez.yerania@tamu.edu

***Abstract*—Machine Reading Comprehension (MRC) allows the opportunity for a system to understand a given context and provide an answer based on query on this context. In this paper, we explore a variety of methods that have been developed to provide high F1 and exact match scorings on the SQuAD data set with the goal of implementing a multi-layer model that combines a variety of these features. We provide flexibility in configuring a number of parmeters in order to analyze the different configurations of networks. Our evaluation and results demonstrate that our best model is a combination of GLoVe embeddings, an LSTM approach to RNN, a BiDAF weight attention, and a smart span method to provide the best answer possible.**

I. Introduction

Machine Reading Comprehension (MRC) provides the ability for computers to read and understand natural language text and it can be considered as one of the necessary abilities for artificial intelligence [1]. In order to train a system in answering a query about a given paragraph, the model will need to be able to create complex interactions and connections between the context provided and the query. Being able to extract such information definitely has its caveats, but could be beneficial for a number of domains, including improving search engines in documents that are domain-specific and providing support when answering customer service inquiries. Systems from previous works have achieved promising results, but are typically characterized by using attention weights on summarized context, the attention weights are computed and dependent on previous steps, and attention layers are computed unidirectional, usually from query to context.

In this paper, we focus on exploring and analyzing a variety of networks in order to find the best model to provide an adequate answer to the query given. The generic idea of our network is to provide an embedding layer, an encoder layer, an attention layer, and the final output layer. Each of these layers has a variety of features that were evaluated, including comparing word and character embeddings, the addition of highway layers, comparing Gated Recurrent Units (GRU) and Long Short Term Memory (LSTM) performance, using a unidirectional attention layer versus a Bi-Directional Attention Flow (BiDAF) layer, and finally using softmax to predict the span based on maximizing the probability distribution. Each of these features has demonstrated promising results individually and therefore, we combine a number of these features in order to develop the best model to solve the MRC challenge. Our final model is able to outperform a number of previous approaches from the SQuAD test set leader board. The rest of the paper is dedicated to describing a few previous approaches, further details on the data set we used and the approach we took for each layer, the evaluation metrics we used, and the final results for the number of models we used along with future work for this task.

II. Related Work

In order to tackle the MRC challenge, a major contributor to developing these models has been the number of large datasets that have been developed and made available. In 2013, MCTest was too small of a dataset in order to be able to train an end-to-end model [8]. In the recent years, industry has provided additional attention to artificial intelligence and as result machine comprehension has become a strong factor in understanding natural text, which lead to CNN/DailyMail dataset in 2015 [5]. However, by 2016, the Stanford Question Answering Dataset (SQuAD) was published providing an extensive dataset with a large number of questions, answers, and a wide-ranging of topics [6].

Typical end-to-end architectures have used a variety of attention mechanisms, one of which depends on the previous step for attention weights. Bahdanau et al. uses this approach referred to as dynamically updating the weights [3]. However, Hermann et al. and Chen et al. demonstrate that the accuracy of the model can increase if a bilinear term is used in order to compute the weights [5]. Furthermore, Kadlec et al. feeds the attention weights after only computing them once into the output layer without depending on previous weights [7]. Each of these approaches are only unidirectional, where the answer is derived from the correlation between the query to context and consider the answer is a single token, which is not appropriate for the SQuAD data set. The BiDAF research paper describes the ability of performing an attention layer from context to question and question to context, which demonstrates a better performance than the previous unidirectional approaches [9]. In addition, Wang & Jiang explored an LSTM architecture for natural language inference (NLI) and demonstrated how the network provides emphasis on word-level matches and relationships between words [12]. Inspired from LSTM networks, highway layers have proven to overcome the difficulty of training a network the deeper it increases considering deepness is crucial to a network's success [10]. Based on these features that have demonstrated improvements to current machine comprehension tasks, we

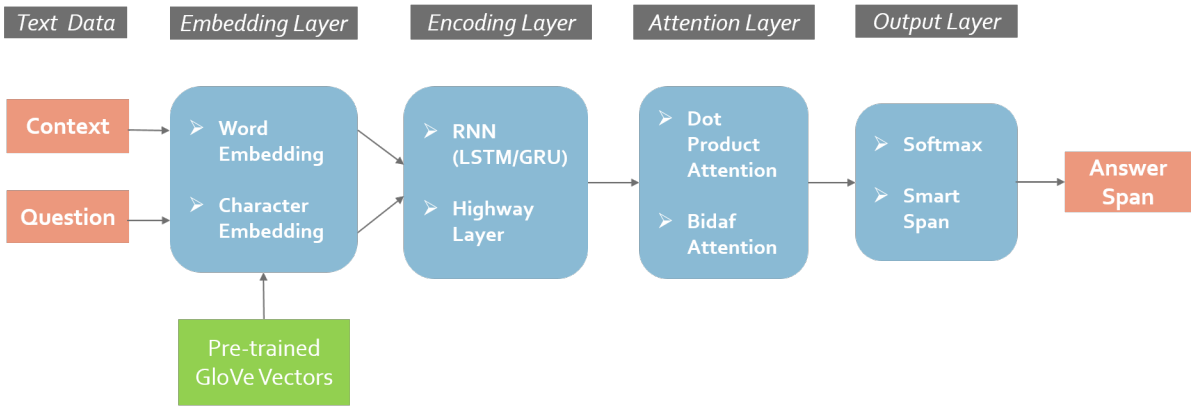


Fig. 1. An overview of the model approach and features implemented within each layer

implemented a model that provided flexibility in using a combination of these aspects in each layer.

III. Approach

Our MRC model is a multi-layer network that consists of four layers as shown in Figure 1: embedding layer, encoding layer, attention layer, and output layer. Each of these layers has a variety of features and complexity that we have implemented based on previous works in order to improve the performance of the model.

A. Dataset Analysis *[Yerania Hernandez]*

The data set used for this model was the Stanford Question Answering Dataset (SQuAD). This data set was created with the help of crowd workers on a set of Wikipedia articles. The questions were asked by the crowd workers and the answers can be found in a given segment of text. With over 100,000 question and answer pairs, it provides our team a large enough data set to train and test our model. Unlike other data sets, SQuAD provides a diversity of questions considering it is larger than most existing data sets and it does not take the simple approach of just providing answer choices from where the machine picks from. Instead, the system needs to use logical inference in order to answer the questions from the passages provided. Considering SQuAD uses Wikipedia articles, the data set is clean compared to other document systems, having no spelling mistakes and the answer is always in the passage.

With further analysis on the train data set, we were able to select our hyperparameters when initializing our network. In the train context data set, we discovered that 95th percentile of the distribution in the context length was 245 words and 99th percentile of the context length is 325 words. As a result, our context length for the network was 300 words. In the train question data set, we observed that that the distribution of the length of questions ranged from 18 to 23 in the 95th and 99th percentile, respectively. However, when analyzing the question length with respective to the number of questions, the majority of questions range from 20 to 40 words. As a result, our question length for the network was 30 words. In the train answer data set, we further analyzed the length of answers and the distribution of the answer span based on the start index. This provided further insight in selecting an answer span length for our final output considering the 95h percentile and 99th percentile of the distribution was between 11 to 21 words. In order to obtain this final answer, we further describe our output layer in later sections.

B. Preprocessing *[Yerania Hernandez]*

Before adding and describing the layers we created for our model, we filtered through the dataset in order to separate the context, the questions, the answers, and the answer spans into separate files. This preprocessing aspect of our dataset was essential in order to standardize all the data, such as additional symbols or punctuations and the aligning the indexes of the span with the actual context. After verifying that the answer from the context based on the answer spans correlates with the actual answer provided from the documentation, we were able to create tuples of the context, question, answer, and answer span. Each tuple was further divided into its respective file for further usage in the actual network.

C. Embedding Layer *[Stuti Sakhi]*

The first layer in the model is the embedding layer. The embedding layer converts text into numerical vectors. This step is crucial because most of the machine learning techniques, including Neural Networks require numbers as input in order to perform any sort of job. There are various techniques existing for embedding including word embedding as well as character embedding. Word embedding directly convert word to vectors, while character embedding combine the embedding of each character in the word to get the word embedding. Below, we discuss the techniques we used in detail.

1) Word Embedding: Word Embedding is used to generate vector representations for the words in the vocabulary. There are two broad categories of word embedding techniques - frequency based and prediction based. Frequency based techniques decompose the word co-occurrence matrix to derive the vector representations. It majorly relies on the global count statistics to arrive at the vectors. On the other hand, prediction

based techniques take only local context into account and learn word embedding which capture meaning in the vector space. Considering the global count statistics and learning dimensions of meaning both are equally crucial in deriving word embedding. Keeping this in mind, we went ahead with using GLoVe Vectors for the word embedding as it combines the best of both worlds. The GloVe vectors are learned by optimising a loss which uses global count statistics information. We used the pre-trained GloVe vectors of dimension 100 for our project. So each word in the dataset(context and question) was converted to a vector using these pre-trained GLoVe vectors.

2) Character Embedding: Character embedding is another way to convert text to vectors. Here we have a vector for each character in our dataset. To get the word embedding we combine the character embedding of the constituent characters. Character embeddings have less dimension as the number of characters are limited. Moreover, character embeddings help us to utilise the internal structure of the word and also handle out of vocabulary words. We used the character level Convolutional Neural Network(CNN) [9] to learn the character embeddings.

For the character level CNN, we start with representing each character with trainable character embeddings c_1, c_2, ..,c_l. Now each word can be represented as e_1, e_2,.., e_k using it. Now, these word representations are input into a 1 dimensional CNN to get the hidden state embedding of the words.The idea is that, each hidden vector is a combination of a window of characters due to the convolution. Both the embedding and filter for the CNN are learned during the training of this model.

D. Encoding Layer ***[Stuti Sakhi]***

Once we had the vector representation of words, our next step was to make each word aware of the words before it and after it. This is a very important step as words individually only give partial information. It is the sentence which actually holds the complete meaning. To make each word aware of its context, we used a bidirectional Recurrent Neural Network. We also used the highway layer as a part of the encoding in some of the cases to model higher complexity. Now, we discuss each of these in detail

1) Recurrent Neural Network: A recurrent neural network(RNN) which is capable of handling sequential data. The hidden state in the RNN, remembers the previous instances and hence can handle sequential data. Figure 2 shows an unrolled RNN. Here, we can view inputs, X_1, X_2,.., X_n as the word embedding of a sentence (context and answer in or case). Each of the hidden state H_t is found using the present input Xt and previous hidden state H_{t-1}. Equation 1 below shows this relation. W_{hh} and W_{xh} are learned during the training.

$$H_t = tanh(W_{hh}H_{t-1} + W_{xh}X_t) \tag{1}$$

RNN, in theory can remember information which was processed long back. In reality however, they face an issue of diminishing gradient which does not allow them to do so. For this reason we decided to work with two modifications of RNN, Long Short Term Memory(LSTM) and Gated Recurrent

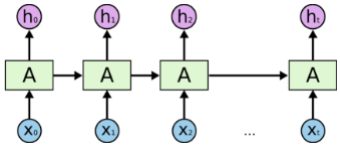


Fig. 2. Recurrent Neural Network

Unit(GRU). Both of these add extra gates to the RNN, thus making it easier for information from past to pass.

A GRU uses an update gate and a reset gate. The update gate decides on how much of information from the past should be let through and the reset gate decides on how much of information from the past should be discarded. On the other hand, LSTM has three gates forget gate, update gate and output gate. Hidden state is computed using the forget gate and update gate. The output gate determines how much much representation the hidden state must have in the output. LSTM and GRU are complex models when compared to a vanilla RNN. However, the added computation allows us to capture information from long sequences. Generally LSTM captures more information from past when compared to GRU. We compared the performance of both of these to find a better fit for out data set.

To make sure words are aware of other words both from left as well as right, we used a bidirectional LSTM/GRU. Bidirectional LSTM/GRU just concatenate the the hidden states of two LSTM/GRU in opposite directions.

2) Highway Layer: Since this is a complex problem, we decided to add more non linearity to the model by adding fully connected layers. However, as our model was already deep, a fully connected layer would not perform well and we might end up losing valuable information from the starting layers. For this reason we decided to add in a highway layer [4]. A highway layer is inspired by LSTM. It has 2 gates transform gate and the carry gate. The transform gate decides how much of the input must be represented by the non linear transformation while the carry gate decides how much of the input must be passed as it is to the next layer. Lets suppose x is the input to the highway layer, then the output y can be written as shown in equation (2). Here, H(tranformed input) is a function of W_h, T(tranform gate) is a function of W_t and C(carry gate) is a function of W_x. All the three W_h, W_t and W_x are learned during the training.

$$y = H(x, W_h).T(x, W_t) + x.C(x, W_x) \tag{2}$$

We tried out various combination in this layer GRU and LSTM with and without the highway layer.By the end of this layer, we have the hidden state vectors for both context and the question.

E. Attention Layer ***[Stuti Sakhi]***

Now that we have both the context and question hidden layers, our next step would be to understand which part of the context is relevant to the question. This layer is called attention as it decides where in the context we need to give attention in

order to answer the given question. Intuitively speaking, this layer will output a vector which assigns weights to each word in the context according to their relevance with the question. We tried out two attention mechanisms, Dot Product Attention and Bidirectional Attention flow in our implementation.We discuss the two in detail below.

1) Dot Product Attention: Dot Product Attention is one of the most primitive attention mechanisms. Lets denote question encodings as q_1,q_2,..,q_M and context encodings as c_1, c_2,...c_N. We evaluate the attention distribution ^{i} for each context state c_i as follows.

$$e^i = [c_i^T q_1 \;\; c_i^T q_2 ... c_i^T q_M] \tag{3}$$

$$\alpha^i = softmax(e^i) \tag{4}$$

Now we take the weighted sum of the question hidden states q_i to get the attention output ai for each context state c_i.

$$a_i = \sum_{j=1}^{M} \alpha_j^i q_j \tag{5}$$

Then finally we concatenate each of these a_i to c_i to get the final attention output hidden state b_i

$$b_i = [c_i; a_i] \tag{6}$$

There is no learning involved in this Dot product attention.

2) Bidirectional Attention Flow Mechanism: Bidirectional Attention Flow is a high performing attention mechanism [9]. It is based on the idea that attention must flow both ways, from question to context as well as context to question. Lets denote question encodings as q_1,q_2,..,q_M and context encodings as c_1, c_2,...c_N. We evaluate each element S_{ij} of the similarity matrix S such that

$$S_{ij} = W_s^T [c_i; q_j; c_i \cdot q_j] \tag{7}$$

Here W_s is learned during the training. Now, first we perform Context to Question attention. For each context hidden state c_i we evaluate a_i according to the following equations.

$$\alpha^i = softmax(S_{i:}) \tag{8}$$

$$a_i = \sum_{j=1}^{M} \alpha_j^i q_j \tag{9}$$

Now, for the Question to Context, firstly for each i from 1 to N we find m_i. All the m_i are concatenated to get m.Then c', the question to context coefficient is evaluated.

$$m_i = max_j(S_{ij}) \tag{10}$$

$$\beta = max(m) \tag{11}$$

$$c' = \sum_{i=1}^{N} \beta_i c_i \tag{12}$$

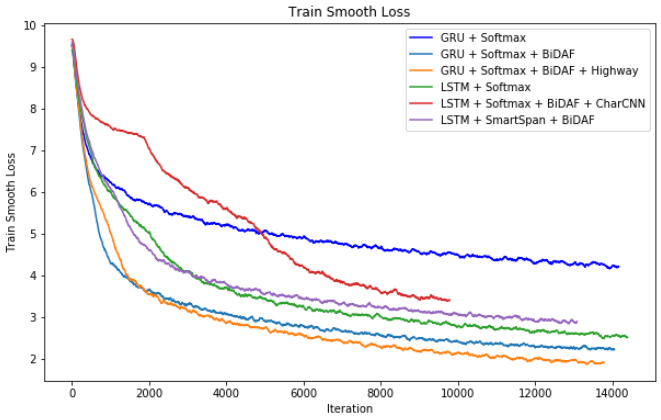


Fig. 3. The loss of the training set for the different models that were configured as the number of iterations increase.

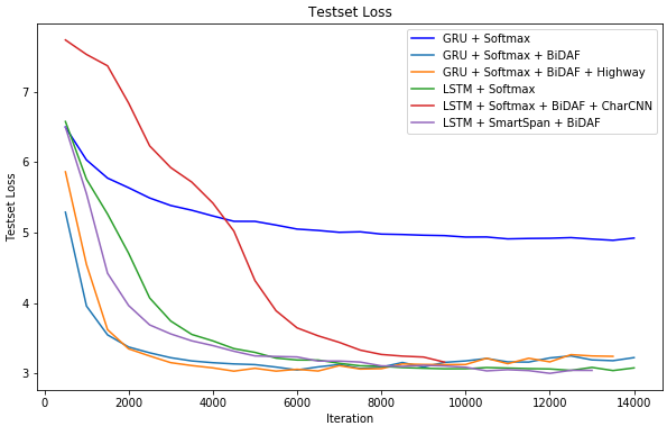


Fig. 4. The loss of the testing set for the different models that were configured as the number of iterations increase.

Finally the Question to Context attention and Context to Question attention are combined to get the final attention out b_i for each context hidden state c_i.

$$b_i = [c_i; a_i; c_i \cdot a_i; c_i \cdot c'] \tag{13}$$

The attention layer takes in question encoding and context encoding and outputs a single vector which is our attention output.

F. Output Layer ***[Yerania Hernandez]***

The final layer of our network consists of what we refer to as the output layer. The main purpose of this layer to predict the span of the answer, from the start index to the end index. From the previous layers, we are able to produce the hidden state layer for the context and the vector for the attention, which is concatenated together as a representation of the context hidden layer and the attention output. This combined layer serves as the input to a TensorFlow fully connected layer. With a fully connected layer, our output layer will be able to compute the start index and end index based on the probability distribution vector produced from a softmax calculation. With these probability distributions, we evaluated

TABLE I
RESULTS FOR SQUAD TRAIN AND TEST SET

Embedding Layer	Encoding Layer	Attention Layer	Output Layer	Dropout	F1 Score		EM Score	
					Train	Test	Train	Test
Glove	GRU	Dot Product	Softmax	0.85	57.41	36.47	45.30	25.95
Glove	GRU	BiDAF	Softmax	0.85	85.01	64.28	72.70	49.11
Glove	GRU + Highway	BiDAF	Softmax	0.85	89.41	65.47	77.30	50.46
Glove	LSTM	BiDAF	Softmax	0.75	81.29	64.83	66.70	50.18
Glove + Char CNN	LSTM	BiDAF	Softmax	0.70	69.07	61.95	55.50	46.89
Glove	LSTM	BiDAF	Smart Span	0.75	77.66	**66.43**	64.90	**50.56**

two different methods in order to obtain the final start and end indexes. The basic method obtains the maximum index probability from the start and end distribution vectors and returns this as the start and end index. The second method, which we refer to as a smart span method, is based on the analysis we had done on our data set. Considering we know that answers span between 11 to 21 words, we selected 15 as the maximum words that the answer should span. As a result, we are able to maximize the probability through the product of the start and end distribution vector in order to obtain the best start and end index. After all these layers have been initialized, we add a loss layer, which simply computes the cross-entropy loss for the start and end index predictions while also using the Adam optimizer in order to minimize the loss across each batch.

IV. EVALUATION [STUTI SAKHI]

We use two evaluation criteria to evaluate our model.

1) Exact Match (EM) : It is a binary measure which checks if the predicted answer matches exactly the actual answer.
2) F1 : F1 is a harmonic mean of precision and recall. Precision the percentage of words in the predicted answer which are present in the actual answer. Recall is the percentage of words from the actual answer which are present in the predicted answer.

V. RESULTS [YERANIA HERNANDEZ]

Our entire network was developed with a variety of parameters in order to allow us the flexibility of testing different configurations. These different networks were implemented using Tensorflow 1.11 and Python 3.6, while training on Google Colab due to the available Tesla K80 GPU. The results of the different model configurations we experimented with are summarized in Table I. As shown, the model with the best configuration uses GLoVe vectors, the LSTM network in the encoding layer, BiDAF attention in the attention layer, and the smart span method in the output layer. The F1 score of this model on the test set is 66.43 and the EM score is 50.56, which performs comparable to the scores displayed on the leader board of the SQuAD data set. In addition, Figure 3 demonstrates the loss of the train set of each of the configurations as the number of iterations increases. The most basic configuration consists of using the GRU network with a softmax as part of the output layer. As the graph displays, the loss is much higher than any of the other configurations. Furthermore, this loss is also replicated in the test set, as shown in Figure 4. All the other configurations demonstrate low losses along with reaching this stage at a much quicker pace. Each of these configurations were run from ten epochs to fifteen epochs, causing them to run for over 14,000 iterations. However, after a certain number of iterations, it is visible that the loss converges.

In addition to the different features we explored within each layer, we also analyzed the effects of certain hyperparameters. After training a few configurations, we observed that accuracy scoring between the train and test had a difference of over twenty points. This concerned us due to the fact that it could be possible that our network was overfitting the results. As a result, we began decreasing the dropout ratio in order to reduce this difference between the training and test set. As is shown in Table I, the decrease in dropout reduced the difference between the train and test set to about ten points. This difference seemed more reasonable considering these results were more correlated to each other and therefore we used 0.75 as the dropout ratio for our final model. In addition, we analyzed the decrease of the learning rate and the effects it had on the scoring as well. However, we only had the opportunity to configure two different learning rates and although it did increase the scoring of our configurations, we cannot make a complete conclusion on the actual effect it had on our results. Our final model used a 0.0008 learning rate instead of the initial 0.001 used for the base model considering previous works focused on the importance of having a low learning rate.

VI. Conclusion

In this paper, we focused on exploring a variety of approaches to develop an MRC model to analyze the SQuAD data set. Our results demonstrate that our best model uses a combination of GLoVe word embedding, LSTM network, BiDAF attention, and smart span to finalize the appropriate answer based on the query and context provided. The analysis on these different configurations provides an insight on the different aspects that have been taken into account to solve the MRC challenge. As a result, we obtained a model that performed similar to some of the methods published on the SQuAD data set leader board. Future work consists of exploring hyperparameters and other configurations, which could possibly increase the F1 and EM score. In addition, due to the flexibility we add to our model, we could further extend this model to implement other types of recurrent neural networks.

References

[1] "2018 NLP Challenge on Machine Reading Comprehension." *2018 NLP Challenge on Machine Reading Comprehension*, 2018, mrc2018.cipsc.org.cn/.

[2] Dwivedi, and Priya Dwivedi. "NLP - Building a Question Answering Model – Towards Data Science." *Towards Data Science*, Towards Data Science, 29 Mar. 2018, towardsdatascience.com/nlp-building-a-question-answering-model-ed0529a68c54.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

[4] Fleming, Jim. "Highway Networks with TensorFlow – Jim Fleming – Medium." *Medium.com*, Medium, 29 Dec. 2015, medium.com/jim-fleming/highway-networks-with-tensorflow-1e6dfa667daa.

[5] Karl Moritz Hermann, Tomas Kocisk y, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, 2015.

[6] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *In EMNLP*, 2016.

[7] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. *In ACL*, 2016.

[8] Richardson, Matthew, et al. "MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text." *2013 Conference on Empirical Methods in Natural Language Processing*, 2013.

[9] Seo, Minjoon, et al. "BI-DIRECTIONAL ATTENTION FLOW FOR MACHINE COMPREHENSION." *ICLR 2017*, 2017, doi:https://arxiv.org/pdf/1611.01603.pdf.

[10] Srivastava, Rupesh Kumar, et al. "Training Very Deep Networks." 2015.

[11] Wang, Shuohang, and Jing Jiang. "Learning Natural Language Inference with LSTM." *Cornell University Lab*, American Physical Society, 10 Nov. 2016, arxiv.org/abs/1512.08849.

[12] Wang, Shuohang, and Jing Jiang. "MACHINE COMPREHENSION USING MATCH-LSTM AND ANSWER POINTER." *ICLR 2017*, 2017, doi:https://arxiv.org/pdf/1608.07905.pdf.

Math Question Answering

Tao Ni

Kunping Huang

Yiran Huang

Syntactic Parsing

- According to the thesis:
  - Backward application of an XTOP (extended top-down tree-to-string) transducer
  - Supported by packages like Tiburon (May and Knight, 2006)
  - About 140 states, and 550 engineered rules

Syntactic Parsing

- According to the thesis:

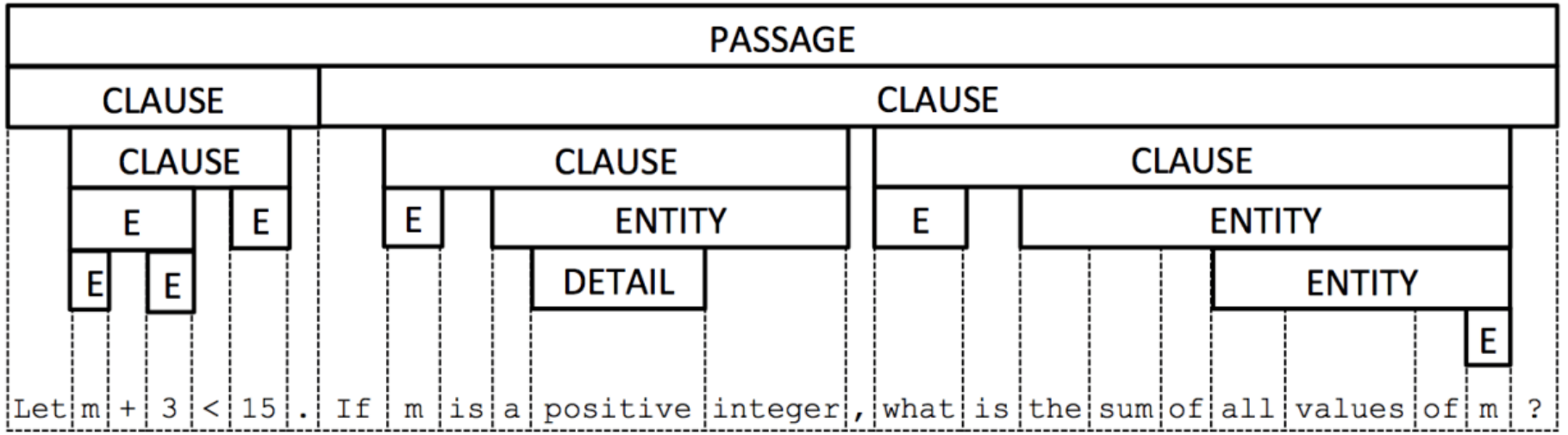


Figure 2: Example syntactic parse. For convenience, we show the correspondence of the nodes of our syntactic parse (top) to the original question passage (bottom). In the parse tree, "E" stands for "ENTITY".

Syntactic Parsing

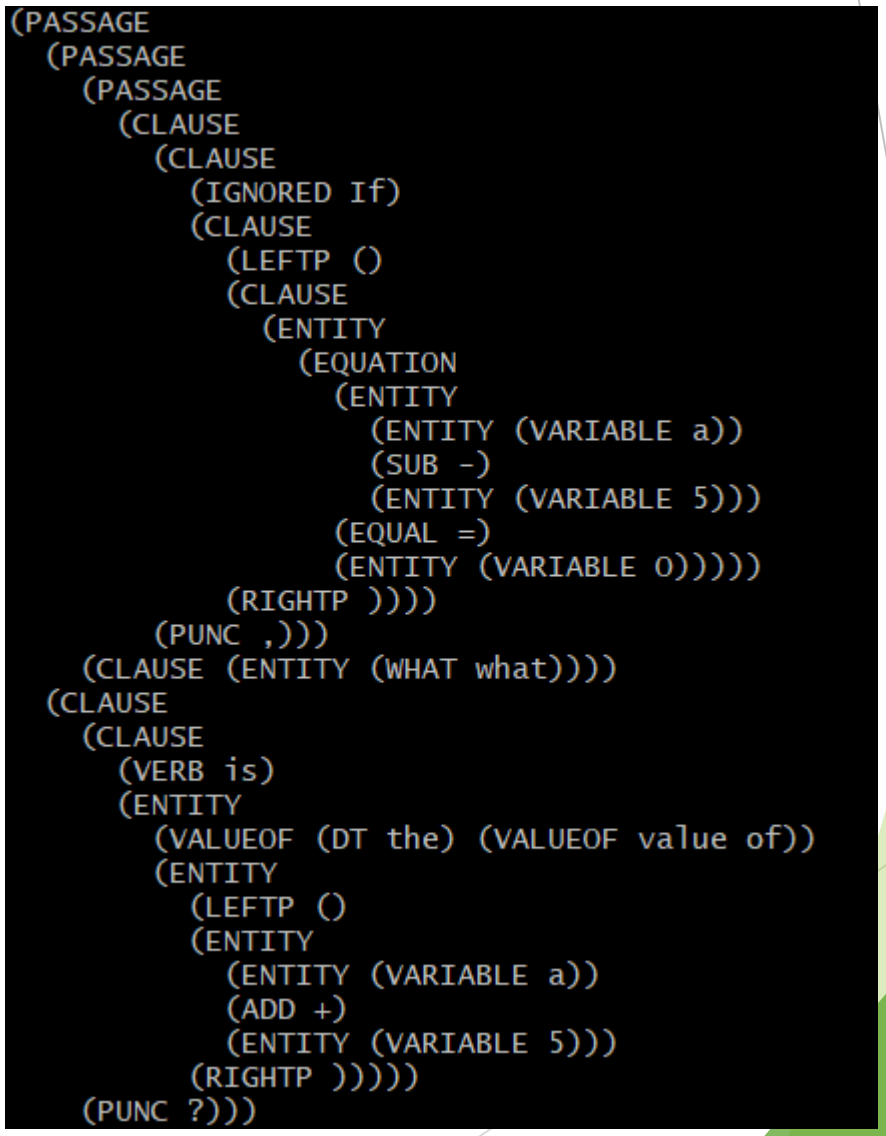
- Our implementation:
  - The Tiburon package is not so easy to use
  - Instead using nltk.parse.chart.BottomUpLeftCornerChartParser
    - This parser can handle left recursions in rules
  - About 50 states and 250 rules

Syntactic Parsing

- Sample results:

Q:

If \(a-5=0\), what is the value of \(a+5\)?



```
(PASSAGE
  (PASSAGE
    (PASSAGE
      (CLAUSE
        (CLAUSE
          (IGNORED If)
          (CLAUSE
            (LEFTP ()
            (CLAUSE
              (ENTITY
                (EQUATION
                  (ENTITY
                    (ENTITY (VARIABLE a))
                    (SUB -)
                    (ENTITY (VARIABLE 5)))
                  (EQUAL =)
                  (ENTITY (VARIABLE 0)))))
            (RIGHTP ))))
        (PUNC ,)))
    (CLAUSE (ENTITY (WHAT what))))
  (CLAUSE
    (CLAUSE
      (VERB is)
      (ENTITY
        (VALUEOF (DT the) (VALUEOF value of))
        (ENTITY
          (LEFTP ()
          (ENTITY
            (ENTITY (VARIABLE a))
            (ADD +)
            (ENTITY (VARIABLE 5)))
          (RIGHTP )))))
    (PUNC ?)))
```

Syntactic Parsing

- Only successfully parses 20% of the questions
  - Problems with word segmentation:

    If \(w+x=5\) and \(y+z=6\), what is the value of \(wy + xz + wz + xy\)?
  - Problems with POS tagging:

    Some of the rules are generated by tagged data (nltk.pos_tag)

    Parsing fails when tagging is incorrect

    e.g. which of the following best describes this relationship?
  - Incomplete grammar rules

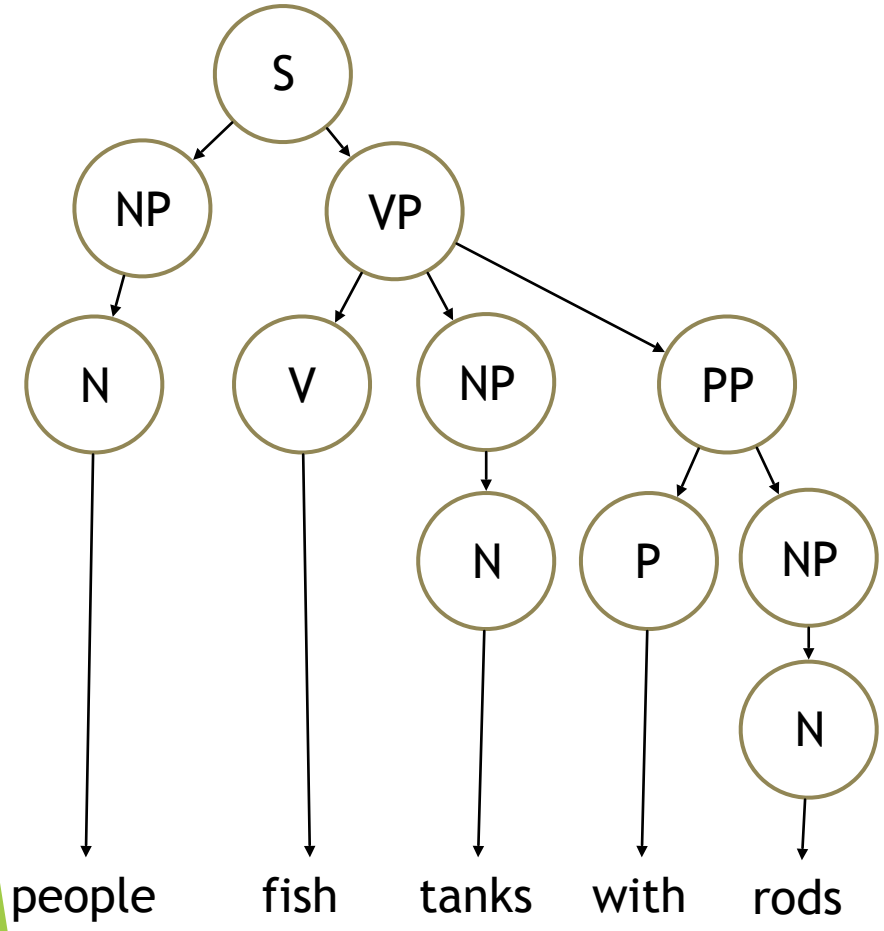
Semantics Parsing

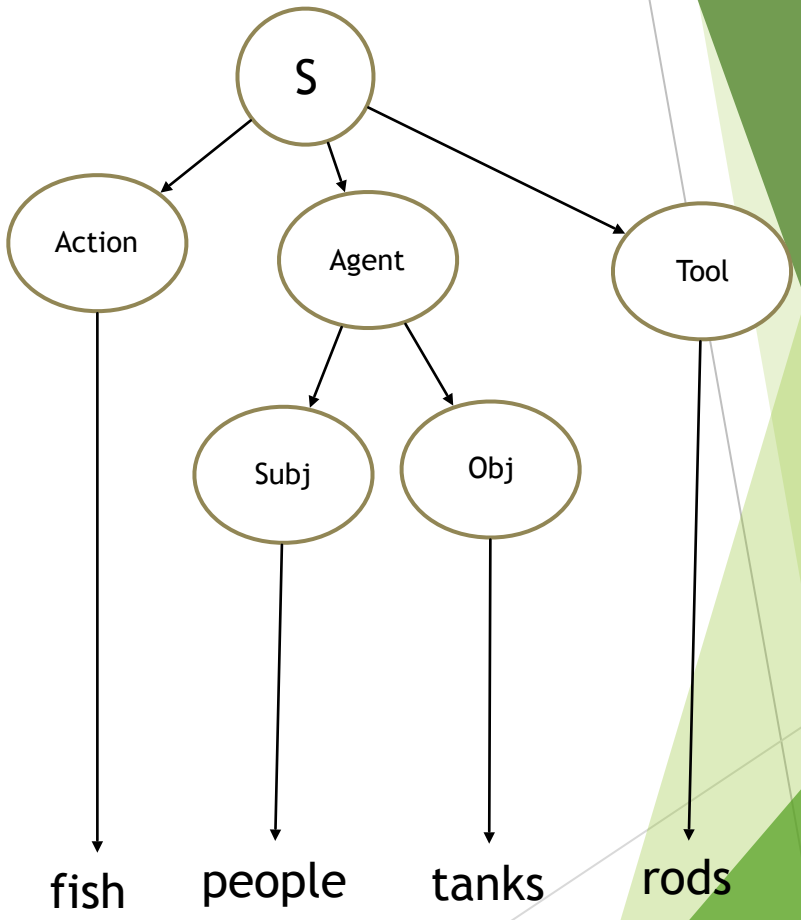
- Multi Bottom-up Tree Transducer
- Semantics Parsing
  - Anaphora resolution
  - Semantic translation

Multi Bottom up Tree Transducer

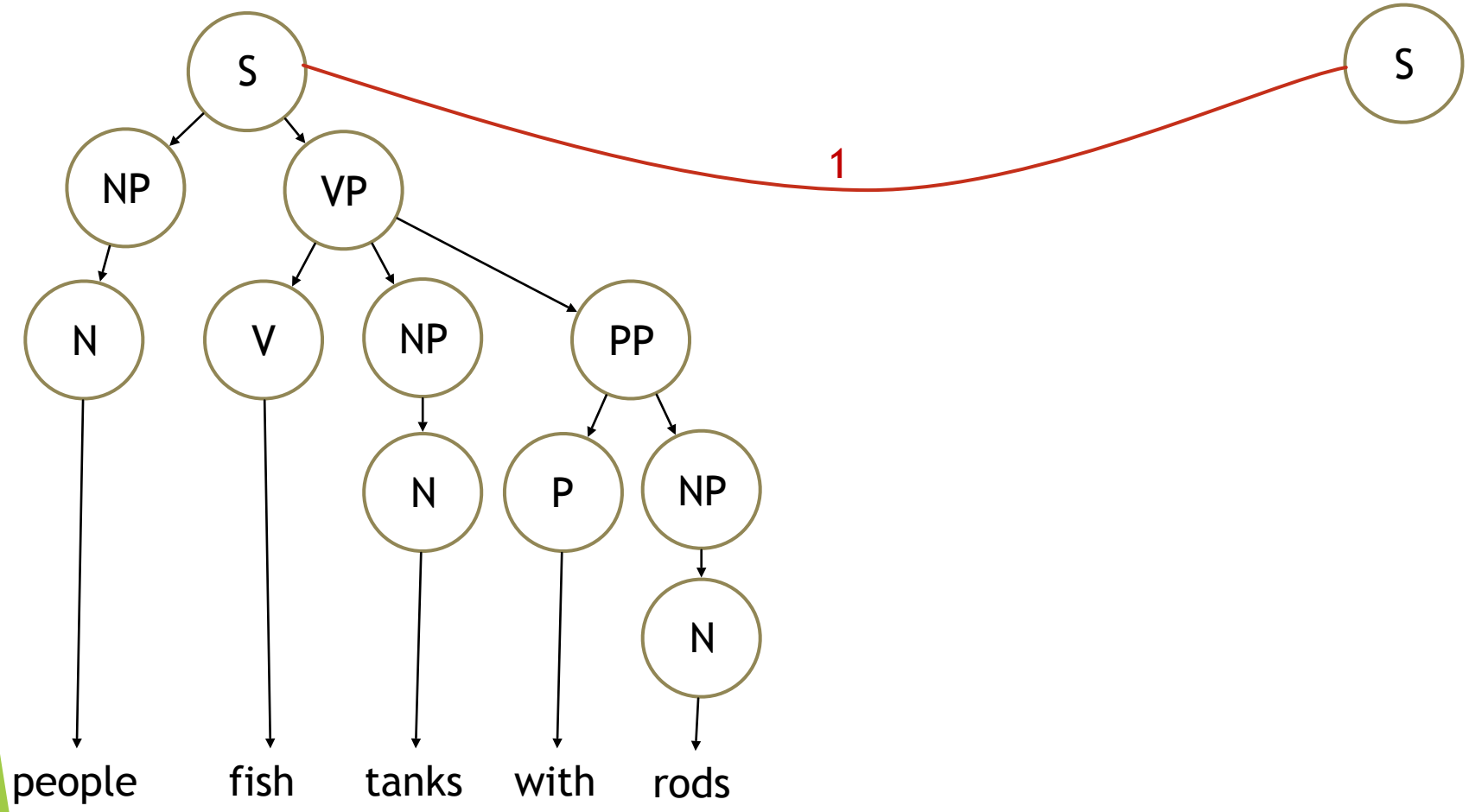
- Math Question: ⟨r, s, t⟩ In the sequence above, if each term after the first is x more than the previous term, what is the average of r, s, and t in terms of r and x?
- The term refers to the term in ⟨r, s, t⟩.
- The typical semantic parser does not work.

Multi Bottom up Tree Transducer

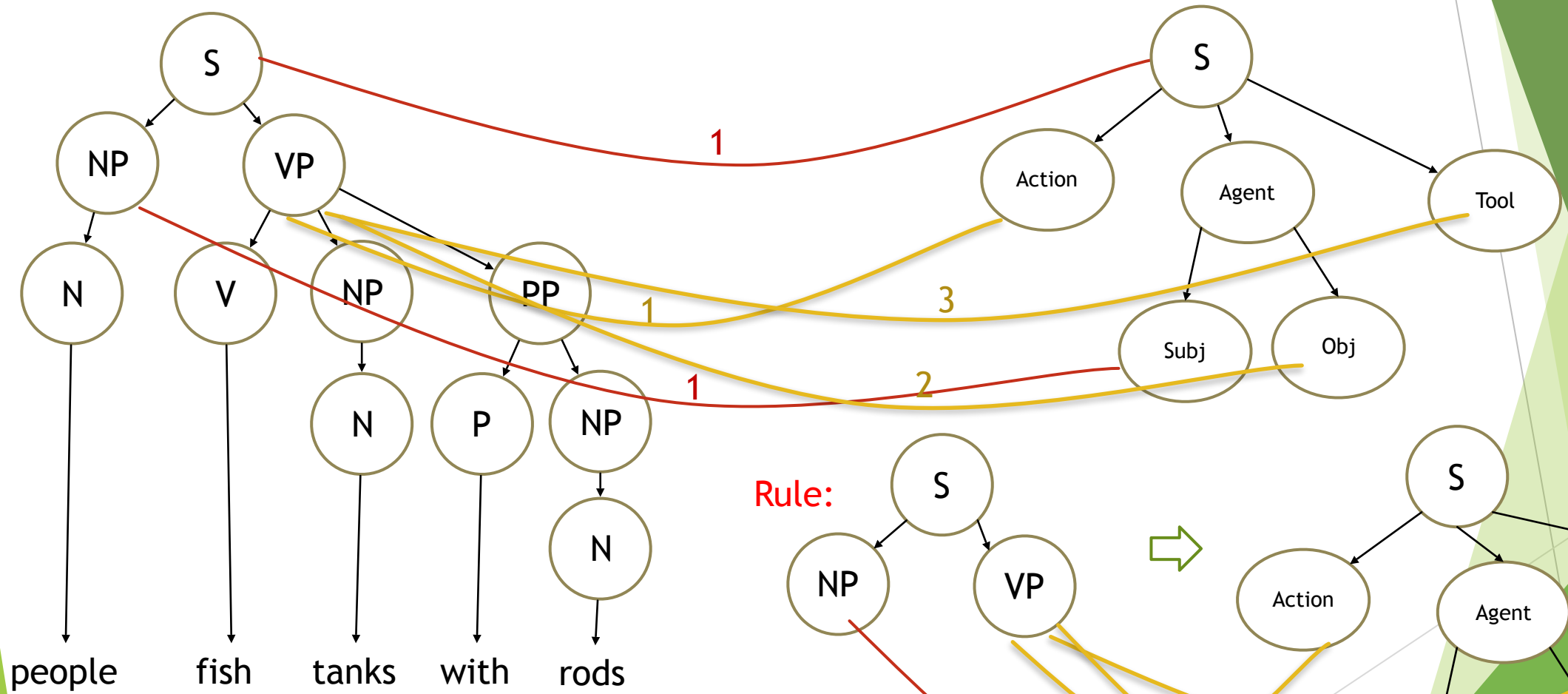




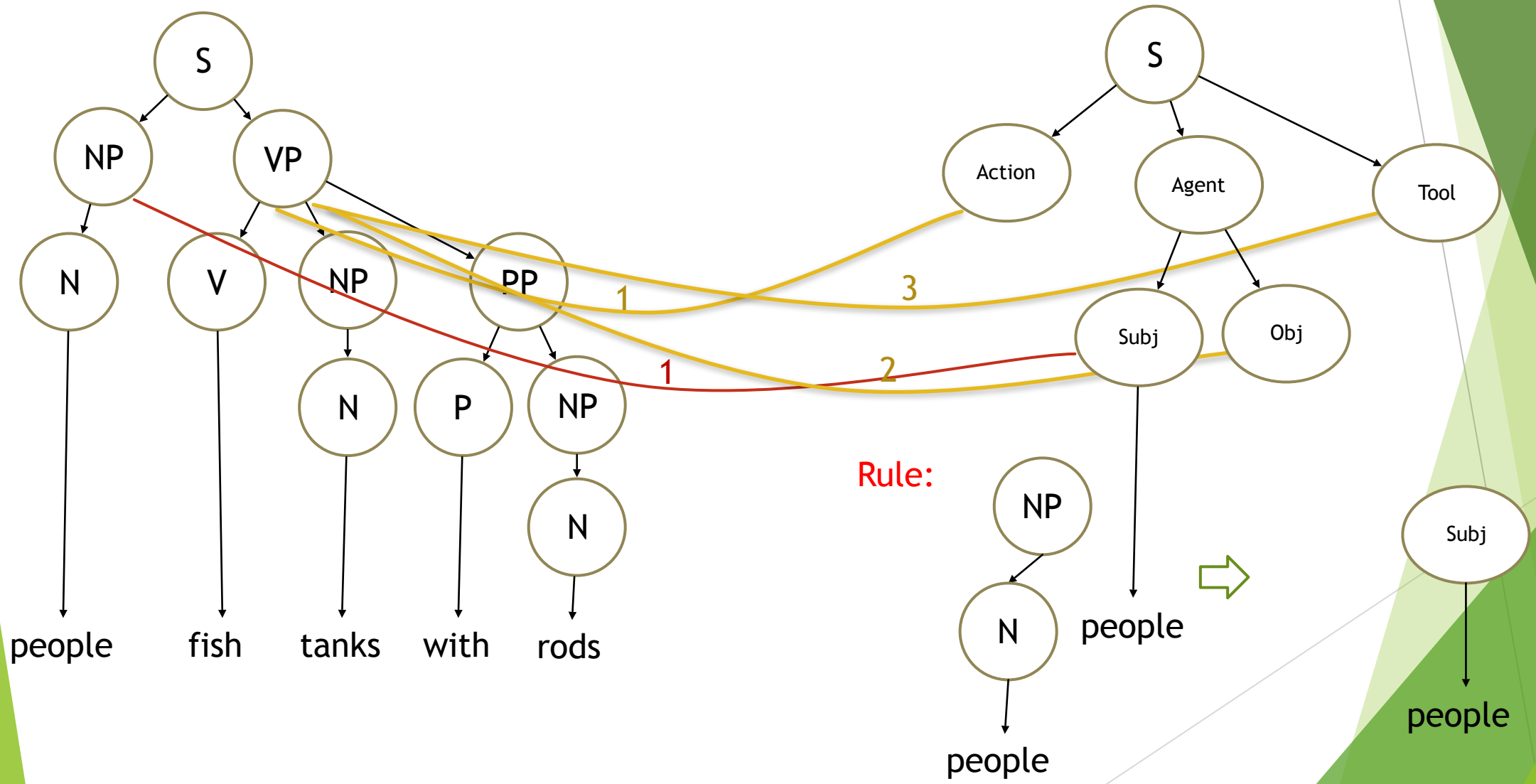
Initial State



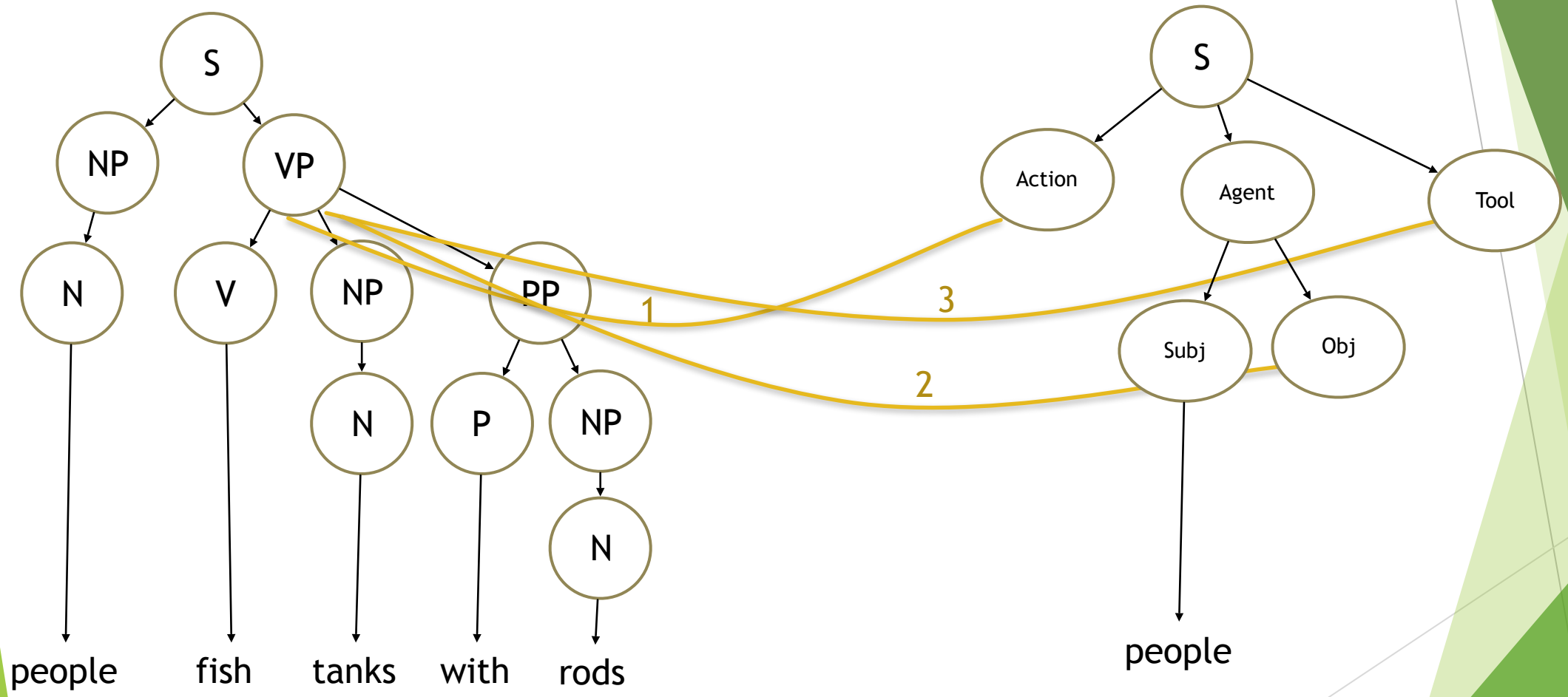
Intermedia State



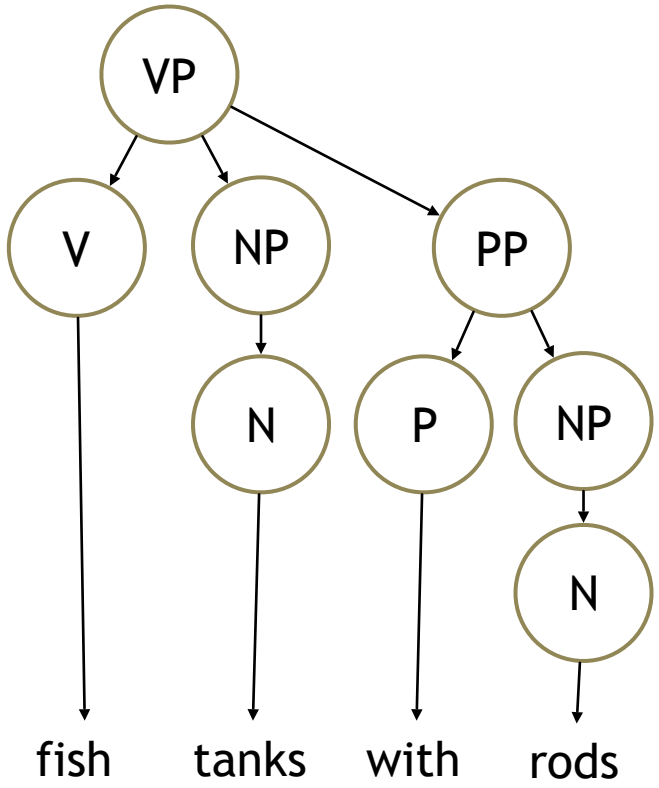
Intermedia State

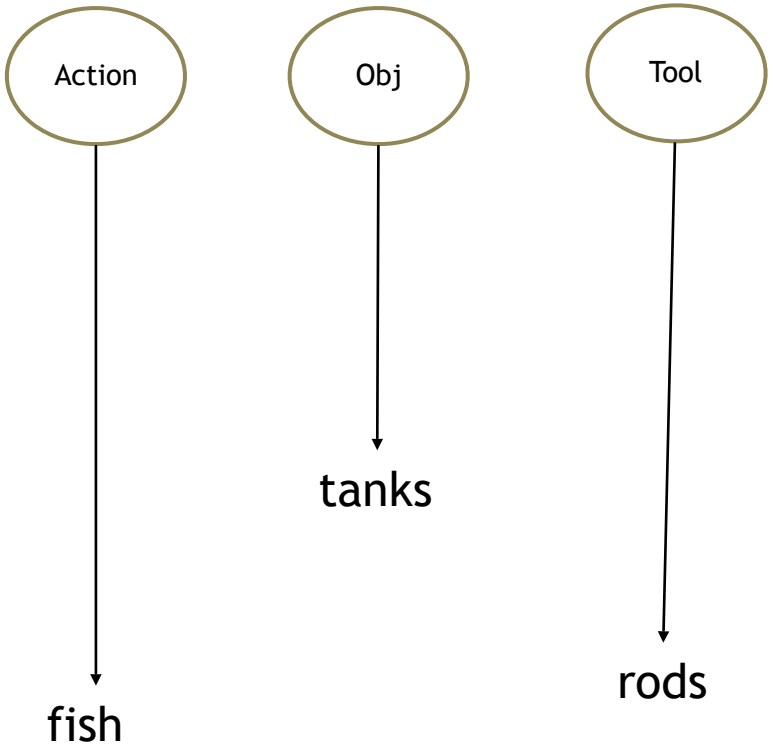


Intermedia State

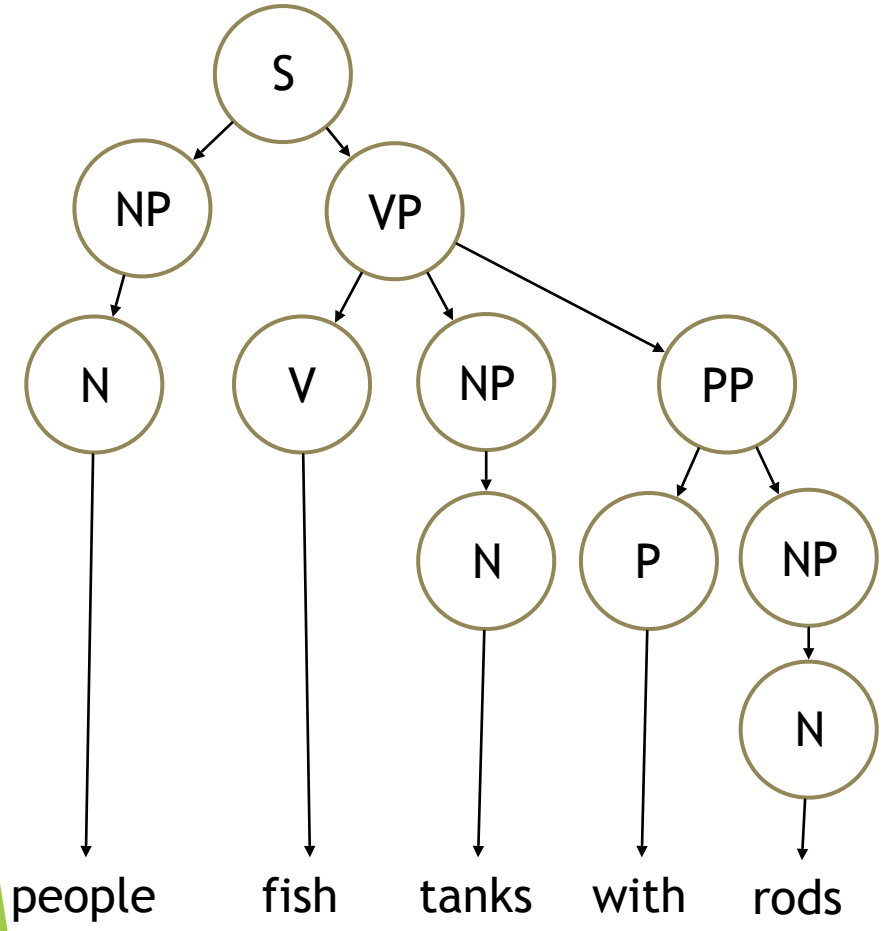


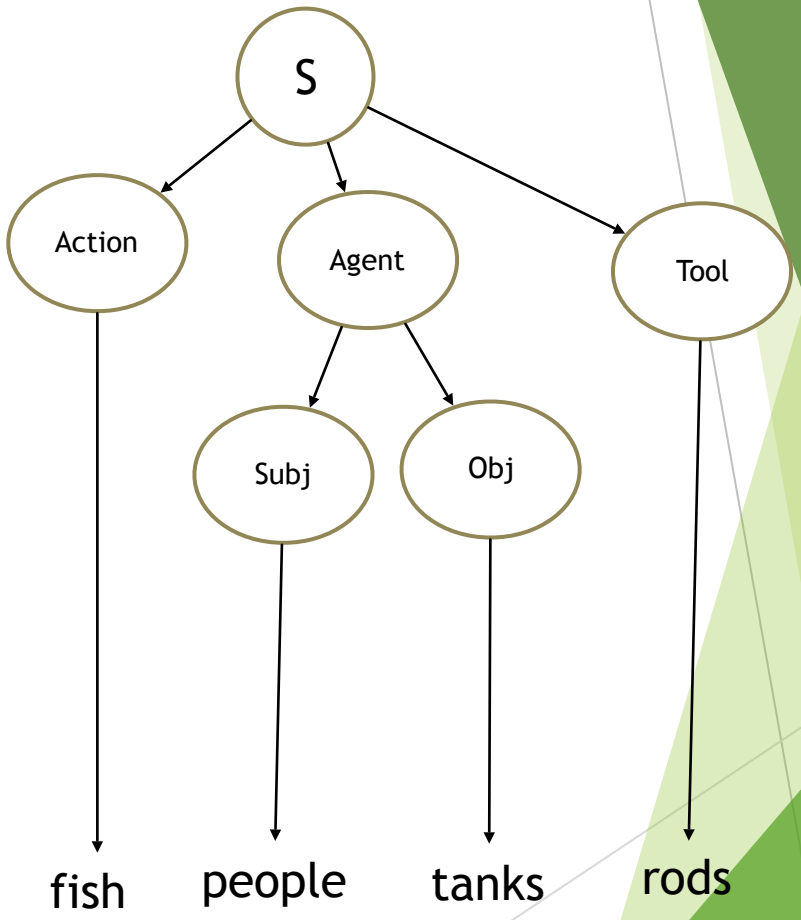
Rule:


VP
V
NP
PP
N
P
NP
N
fish
tanks
with
rods


Action
Obj
Tool
fish
tanks
rods

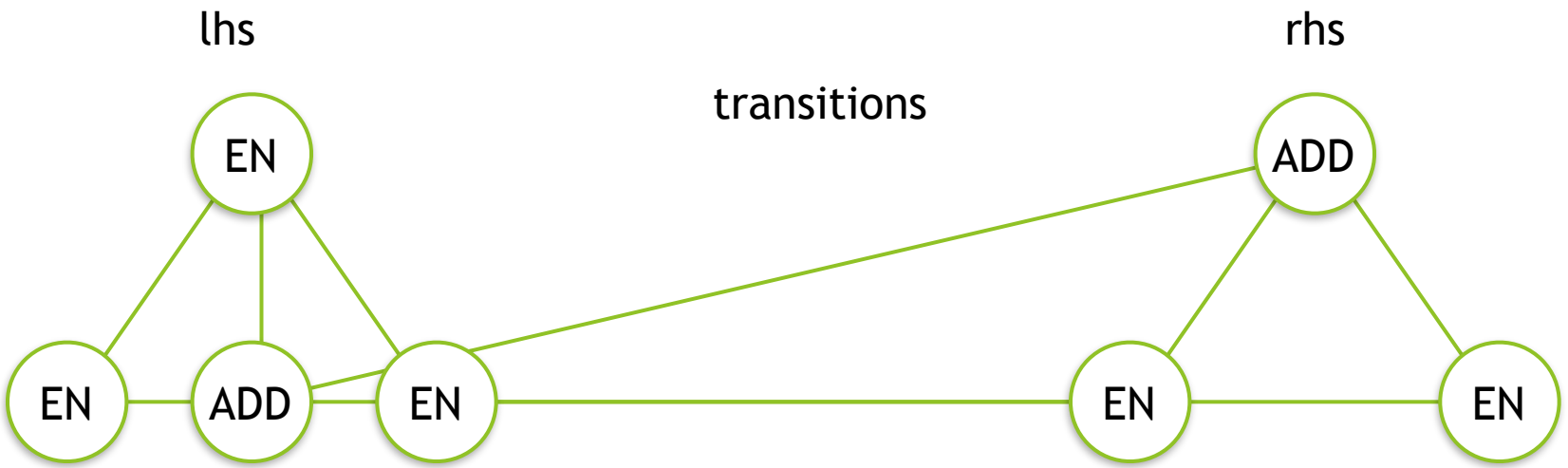
Multi Bottom up Tree Transducer





Tree to Tree Rules

```
- lhs: (ENTITY ENTITY ADD ENTITY)
  rhs:
  - (ADD ENTITY ENTITY)
  transitions:
  - [[0], [[0]]]
  - [[1], [[]]]
  - [[2], [[1]]]
```



Complex Aggregations

- Decomposing Complex Aggregations into Order-independent atoms

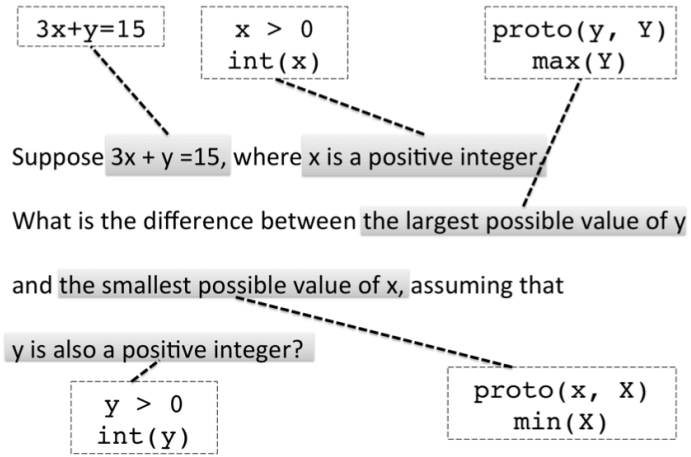


Figure 6: Understanding complex aggregations by decomposing them into order-independent atoms.

Interpretation

- Z3 solver:
  - Z3 is a *Satisfiability Modulo Theories (SMT)* solver.
  - Z3Py: the Z3 API in Python

```
x, y, z = Reals('x y z')
s = Solver()
s.add(x > 1, y > 1, x + y > 3, z - x < 10)
print s.check()
```

  - Cannot handle output directly

Interpretation

- A python script provided by AI2:
  - Question: If 4x + 2 = 26, then what is 4x + 8? (A) 32, (B) 34, (C) 36, (D) 38, (E) 40

```
"id": 10000,
"logicalForm": [
    "(assert (Strategy \"CheckUnsatisfiable\"))",
    "(assert (= (- a 5) 0))",
    "(assert (Not (= ?_id_7_8 (+ a 5))))",
    "(assert (MenuItem \"A\" (- 0 10)))",
    "(assert (MenuItem \"B\" (- 0 5)))",
    "(assert (MenuItem \"C\" 0))",
    "(assert (MenuItem \"D\" 5))",
    "(assert (MenuItem \"E\" 10))"
```

  -

Result

- It is not easy to find the appropriate rules for semantic parsing. So far we haven't worked out a single logical form.

Conclusion

- This study is just the very first step of building a unified system that can solve math word problems. And from this study we experienced the difficulty of building rules as well as semantic parsing. We will focus on building grammar rules more efficiently and making the parsing process more precise in the future.

Q&A

Visual Question Answering

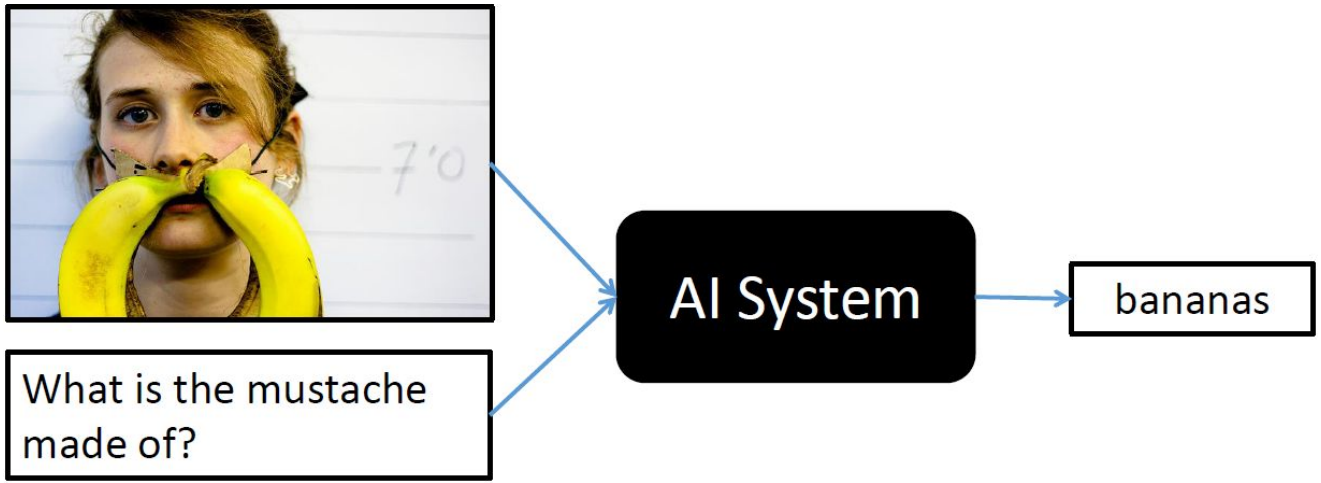
Zhengyang Wang, Yi Liu, Yaochen Xie

CSCE-638: NLP Foundation and Techniques

Dec 4, 2018

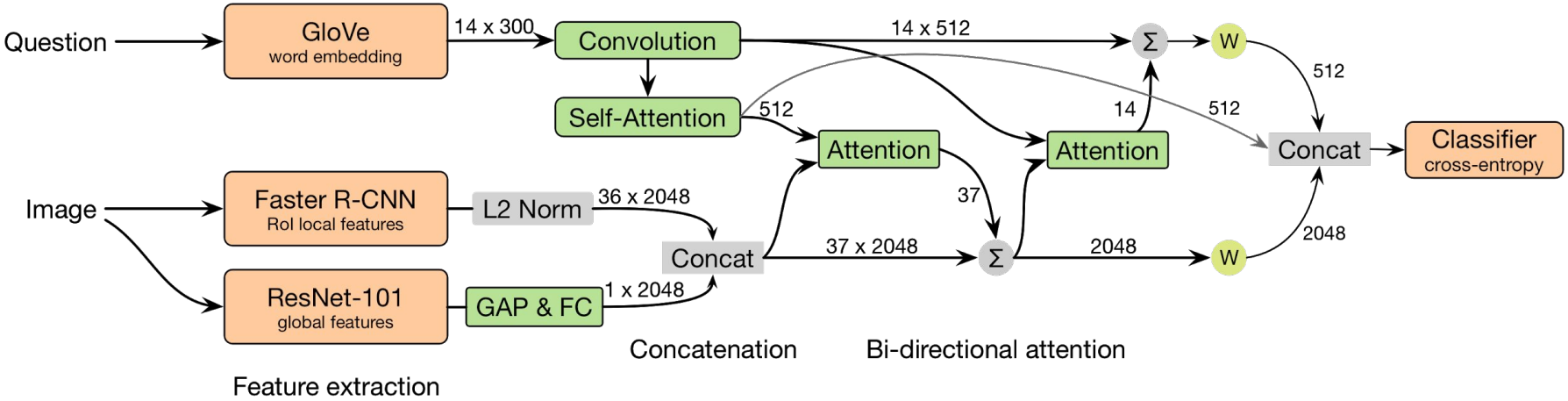
The VQA Task

- Given an **image** and a **natural language question** about the image, the task is to provide an accurate natural language **answer**.

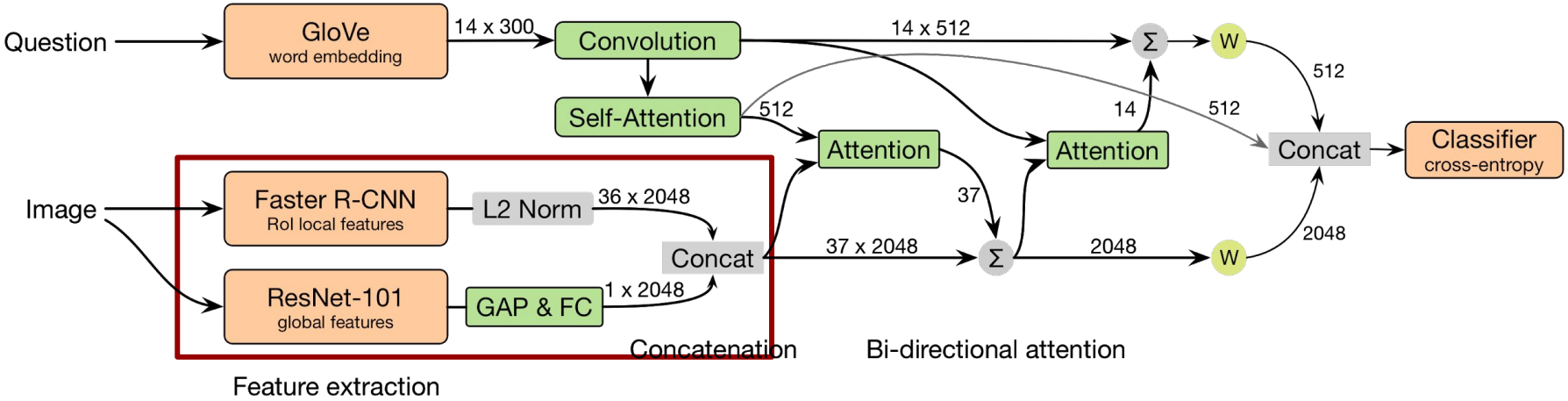


Our Approach

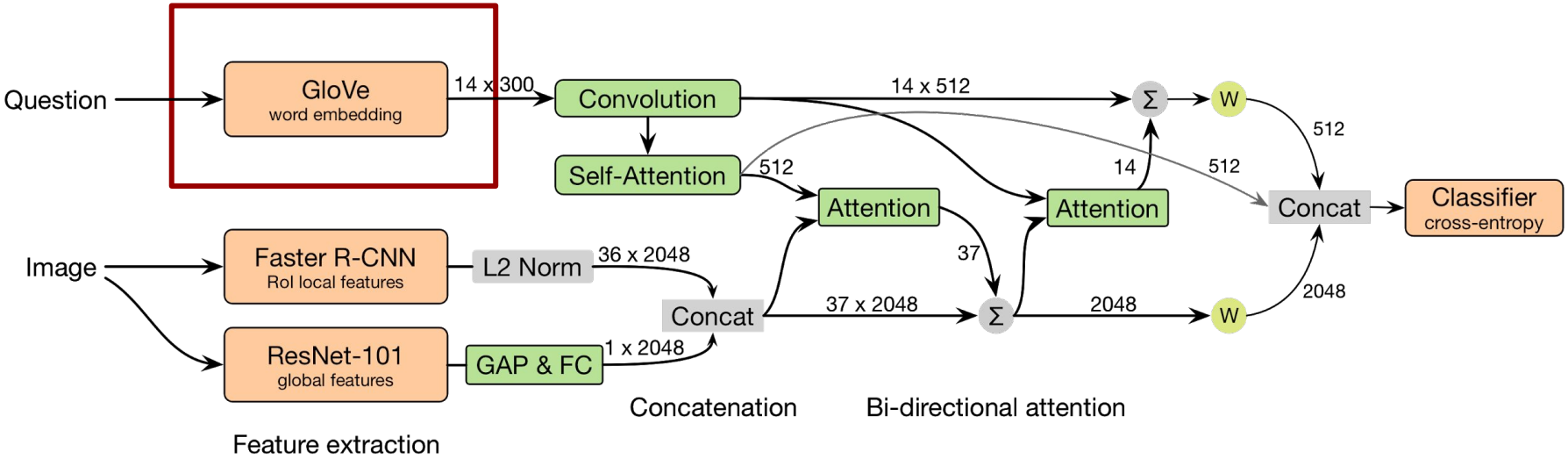
Architecture Overview



Our Approach



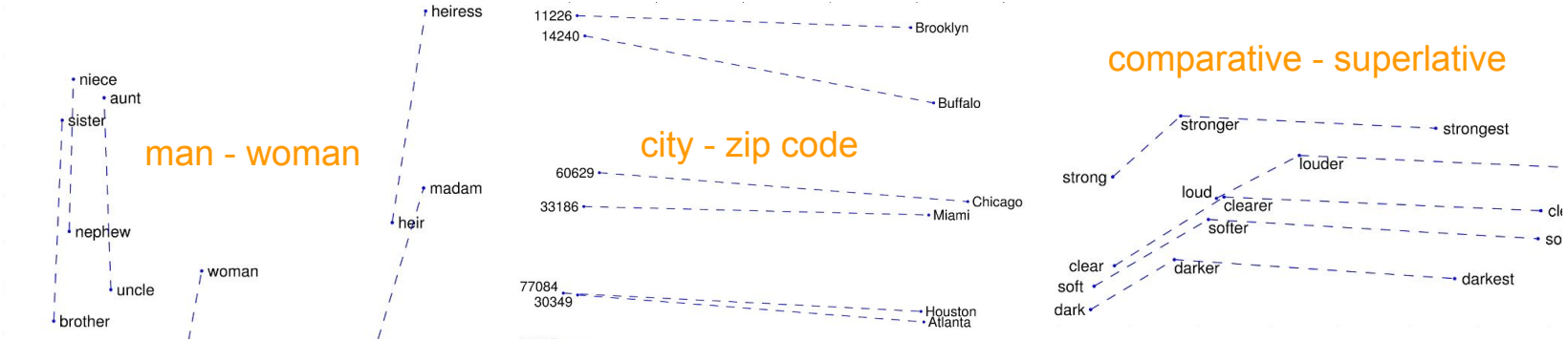
Our Approach



Our Approach

Word Embedding: **word2vec/GloVe**/BERT

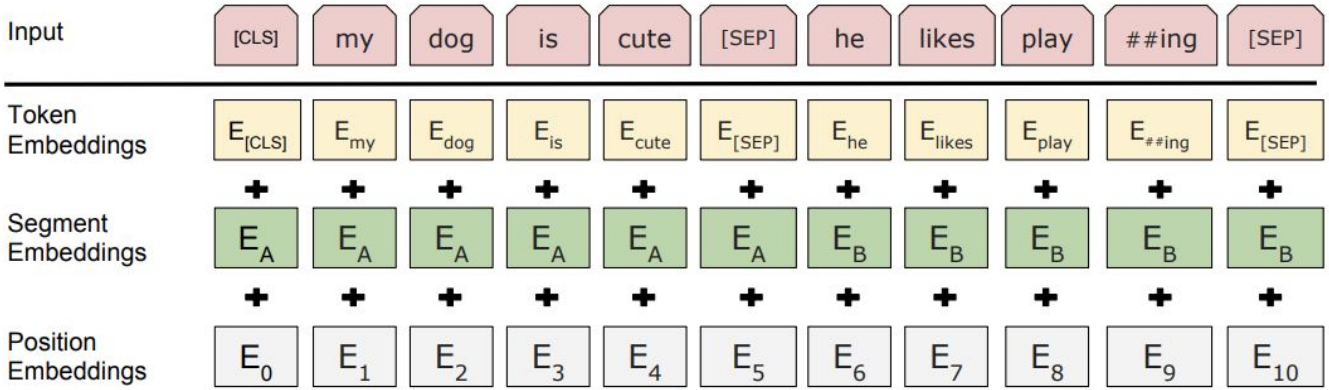
- Euclidean/cosine distance between two vectors measures the similarity of two words
- Linear substructures



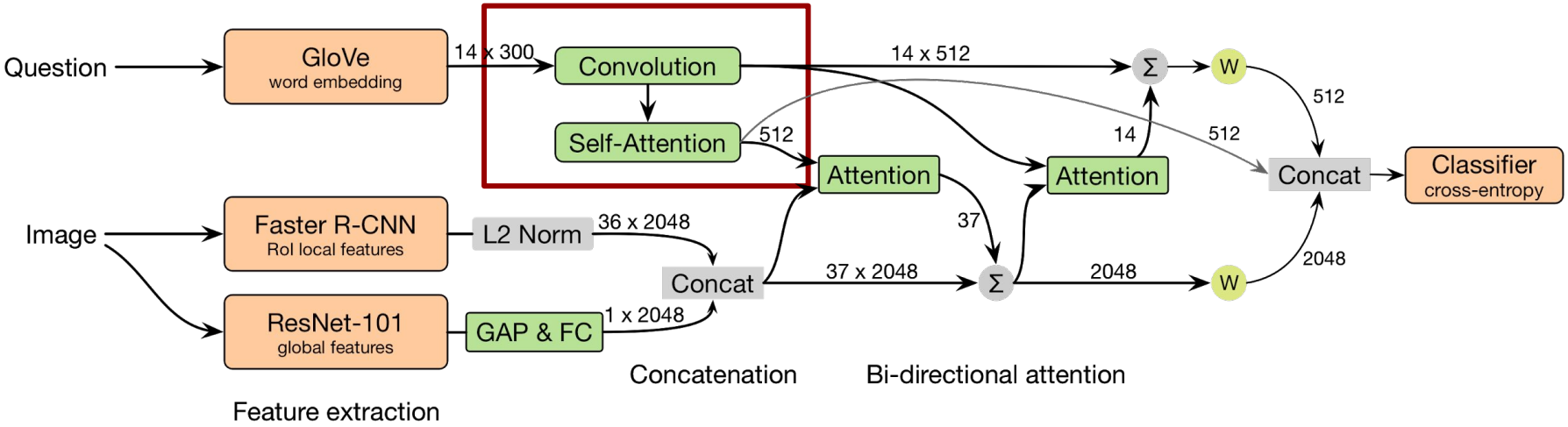
Our Approach

Word Embedding: word2vec/GloVe/**BERT**

- Bi-directional Transformer (attention-based)
- Contextual embedding

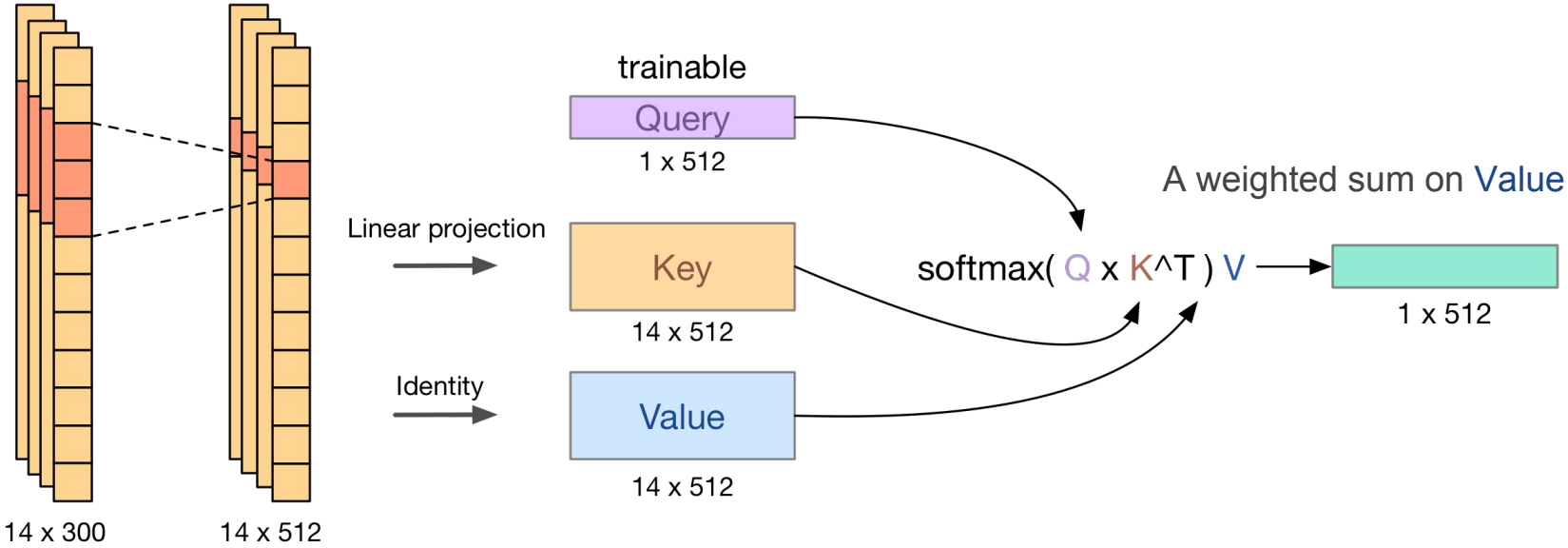


Our Approach

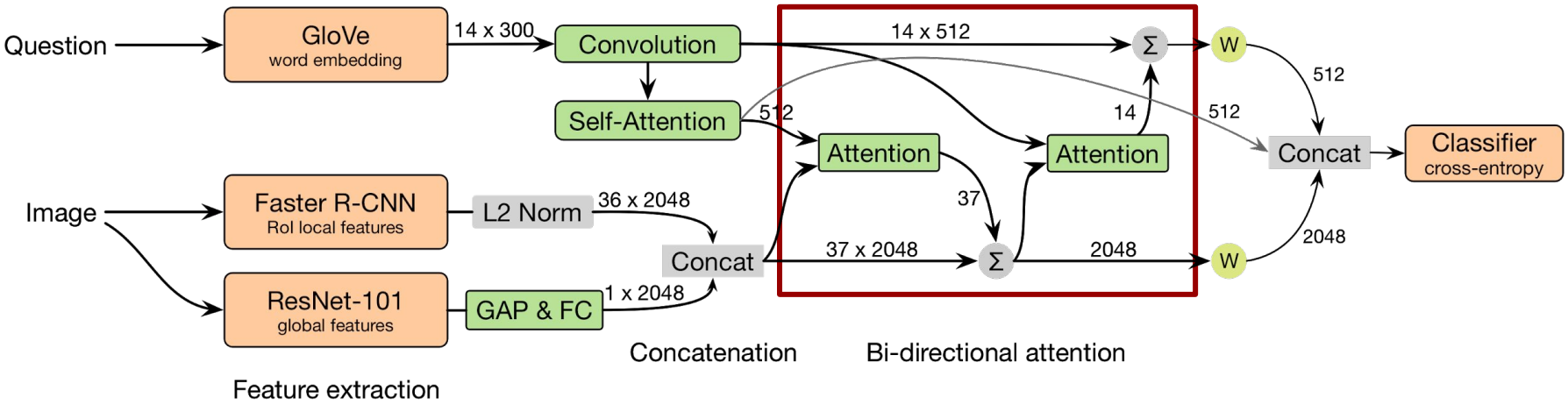


Our Approach

Convolution & (Multi-head) Self-attention

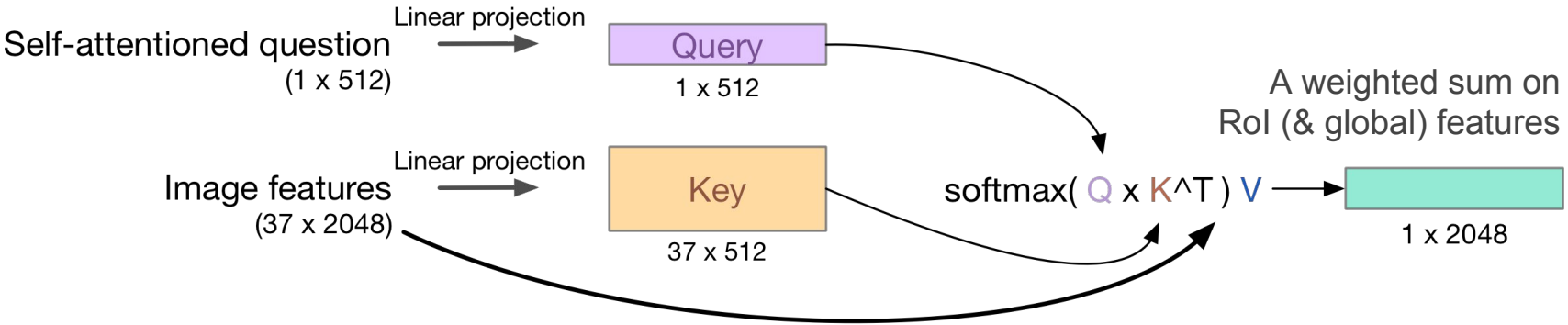


Our Approach



Our Approach

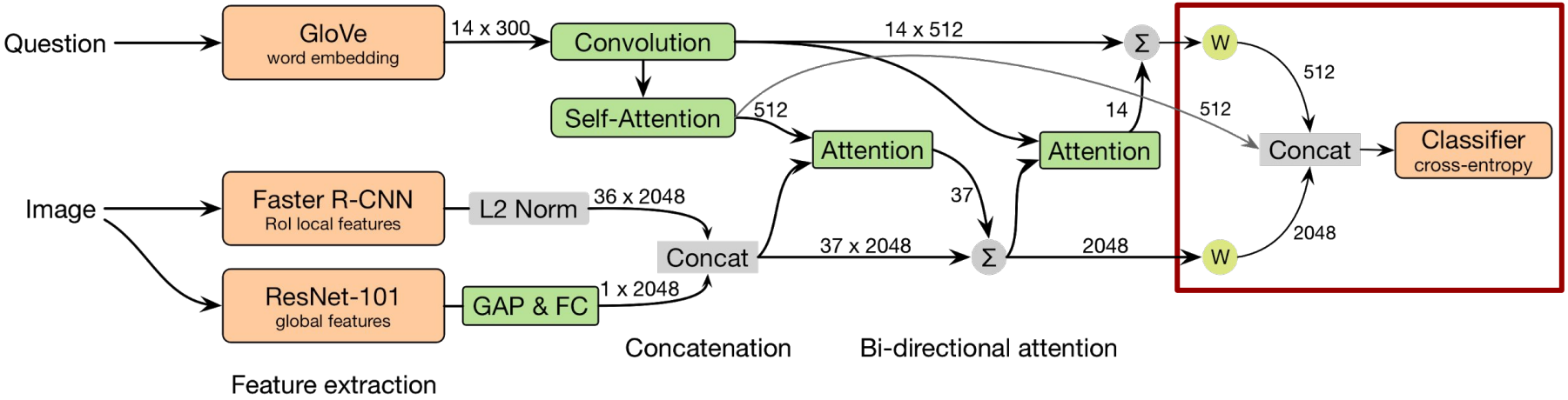
Question-to-image attention



And the **Image-to-question** attention works similarly.

Why do we use **bi-directional** attention?

Our Approach



Dataset & Evaluation

- Dataset we used: **VQA v2.0**

	Images	Questions	Annotations
Training	82,783	443,757	4,437,570
Validation	40,504	214,354	2,143,540

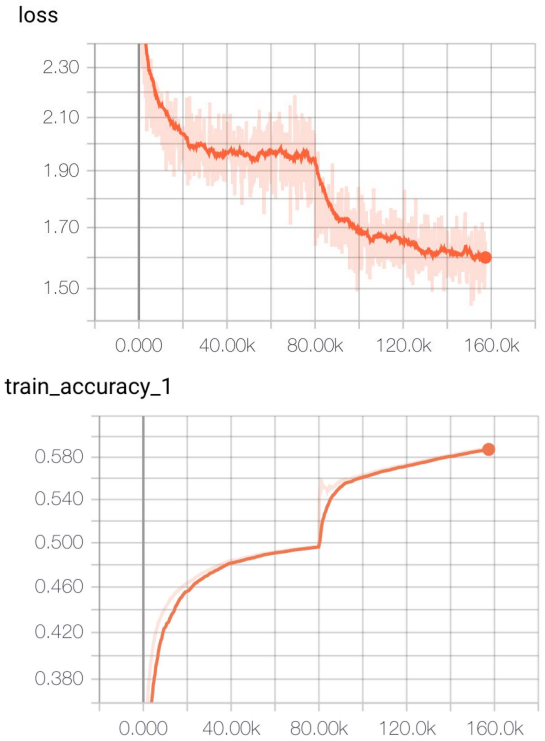
- Evaluation
  - An evaluation metric that robust to inter-human variability in phrasing the answers:

$$\mathrm{Acc}(ans) = \min\left\{\frac{\#\text{humans that said } ans}{3}, 1\right\}$$

Experimental Results

Early stopping at 120k-th iteration to avoid overfitting.

Model	VQA Score
Baseline	58.87
Our model with word2vec	60.43
Our model with GloVe	60.57
Our model with BERT	12.12



Contributions & Remaining Problems

(+) **Attempted to use different pre-trained word embeddings.**

(+) **Extracted the question representation by convolution and self-attention.**

(+) **Proposed bi-directional attention between the image and the question.**

(+) **Used the combination of RoI features and global features to get better performance on specific questions (counting, background, etc.).**

(-) **Still need to explore applying the breakthrough BERT model to achieve a better performance.**

(-) **The classifier part can be improved (e.g., by pre-train)**