

Contents

1	Question & Answering	2
1.1	Question Taxonomy	2
1.2	Question Analysis	3
1.2.1	Watson Q Analysis	3
1.3	Information Extraction (Knowledge Base Population)	5
1.3.1	Introduction:	5
1.3.2	NER	5
1.3.3	Relation Extraction (RE)	6
1.4	QA from Structured KB	9
1.4.1	KBs	9
1.4.2	KBQA	9
1.4.3	Attention	11
1.4.4	Inference in KBs	16
1.5	IR based QA	16
1.5.1	Semantic Search	16
1.5.2	Query Preprocessing	16
1.5.3	Query Formulation	16
1.5.4	Important Papers	16
1.5.5	DataSets	16
1.5.6	Overall Pipeline* NLP Coursera Slides	16
1.6	Answer Sentence Selection	16
1.6.1	LDC Method	16
1.6.2	Important Papers	17
1.7	Machine Comprehension	17
1.7.1	multi-context using seq2seq model	17
1.7.2	Using Memory Networks (from of attention)	18
1.7.3	Using Manually Curated Syntactic/Semantic Features	18
1.7.4	DataSets	18
1.8	Memory Networks:	18
1.8.1	OpenQA with Weakly Embedded Models	18
1.8.2	QA with subgraph embeddings	19
1.8.3	Memory Networks	21
1.8.4	Towards AI-Complete QA	21
1.8.5	End-to-End Memory Networks	21
1.8.6	Large Scale Simple QA	21
1.8.7	Key-Value Memory Networks	22
1.8.8	Recurrent Entity Networks	24
1.8.9	Evaluating Prerequisite Qualities for End-to-End Dialog Systems	24
1.8.10	Dialog-based Language Learning	24
1.8.11	Memory Network Extensions	25
1.8.12	Datasets	26
1.9	IBM DeepQA (Watson)	26
1.10	Chained Reasoning Questions	56
1.10.1	Important Papers	56
1.10.2	DataSets	57
1.11	Semantic Parsing	57
1.11.1	Overview & Important Papers	57
1.11.2	DataSets	58
1.12	Natural Language Inference	59
1.13	Pipeline	59
1.13.1	AskMSR	59

1.13.2	Ephyra	59
1.13.3	Jacana	59
1.13.4	openQA	60
1.13.5	OAQA	60
1.13.6	WatsonSim	60
1.13.7	YodaQA	60
1.13.8	QuASE	60
2	Paraphrase Detection	60
2.1	Taxonomy	60
2.2	Dataset Papers and Datasets	60
2.2.1	Microsoft Research Paraphrase Corpus	60
2.2.2	Other DataSet Papers	60
2.2.3	Datasets	60
2.3	Important Papers	61
2.3.1	ECNU at SemEval 2017	61
2.3.2	Discriminative Improvements to Distributional Sentence Similarity	64
2.3.3	MayoNLP at SemEval-2016	65
2.3.4	Neural Paraphrase Identification of Questions with Noisy Pretraining	68
2.3.5	BiMPM Model	69
2.3.6	more papers	72
2.4	State of Art Papers on MSRPC	72
2.5	Review Paper	73
2.6	Related Papers	73
2.7	Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment	73
2.8	Theory	73
3	Question Generation	74
3.1	Rule based approaches	74
3.1.1	Syntax Based	74
3.1.2	Semantic based	77
3.2	RNN Approaches	90
3.2.1	QG From KB Triples	90
3.2.2	QG from Document conditioned on answer	92
3.2.3	More	92
3.3	Other Approaches	93
4	Misc	93
4.1	Definitions / Summarization	94
4.1.1	Important Papers	94
4.1.2	DataSets	94
4.2	Visual QA	94
4.3	Dialog Generation	94
4.3.1	Benchmark Dataset	94
4.3.2	Learning End-to-End Goal-Oriented Dialog	94
4.3.3	Sequence-to-Sequence Models	94
4.3.4	Spoken Dialog	95

1 Question & Answering

1.1 Question Taxonomy

- A Linguistic Analysis of Question Taxonomies - 2005 - J. Pomerantz

- On local disk
- Towards AI-Complete QA: A Set of Prerequisite Toy Tasks - 2015 - c192
 - <https://research.fb.com/projects/babi/>

1.2 Question Analysis

1.2.1 Watson Q Analysis

- Example:

POETS & POETRY: He was a bank clerk in the Yukon before he published "Songs of a Sourdough" in 1907.
- Focus : part of the Q which is a references to the answer and you can substitute it with the answer and the updated Q will make sense. In the above example, "he" is the focus
 - Used by alignment algorithm which try to align the question with a candidate answer passage, i.e. for proper alignment, the answer in the passage should align with the focus in the question
- Lexical Answer Type: typically the headword of the focus (but not always) and a Q can have multiple LATs (which can be found using co-reference resolution). In the above example "he" is also the LAT as well as "clerk" & "poet"
 - LATs are used to check if a candidate answer type is compatible with the answer type expected by the question
- Question Class: Determines the type of Question, e.g. factoid, definition, reasoning etc. The above example is a factoid question.
 - Q Class is used to determine which algorithms/models are appropriate for processing this question further down the pipeline
- Question Sections: Used in decomposing a question into multiple sub-questions.
 - These are used in decomposition as well as determining constraints an answer must obey.
 - In the above example, "He was a bank clerk in the Yukon" and 'he published "Songs of a Sourdough" in 1907' are the Q Sections

1. Analysis of Questions from the WikiCorpus

- how big is bmc software in houston, tx
 - Focus: How big
 - LAT: big
 - Q Class: Factoid:numerical
- How does the hydrophobic and hydrophilic nature of the phospholipid layer regulate what enters and leaves the cell?
 - Focus: How
 - LAT: description:physical phenomena (from "does NNP regulate")
 - Q Class: Factoid:description
- how do people get color blindness?
 - Focus: How
 - LAT: description:body (from 'how do' and 'get color blindness')
 - Q Class: Factoid:description

- How Works Diaphragm Pump
 - Focus: How works
 - LAT: description:physical phenomena (from 'works')
 - Q Class: Factoid:description
- what part of the pre-world war 1 arms race was the most intense?
 - Focus: what part
 - LAT: part
 - Q Class: Factoid:numerical
- who won the 2010 world cup
 - Focus: Who
 - LAT: winner (from 'won')
 - Q Class: Factoid:location
- who plays dumbledore in harry potter 6
 - Focus: Who
 - LAT: player/actor (from 'plays')
 - Q Class: Factoid:actor
- what is the scientific name of the eastern tiger salamander?
 - Focus: What
 - LAT: scientific-name
 - Q Class: Factoid:scientific-name
- what was the GE building in rockefeller plaza called before
 - Focus: What
 - LAT: called/name
 - Q Class: Factoid:location
- where is the world cup in 2010
 - Focus: Where
 - LAT: country
 - Q Class: Factoid:location
- what is the name of chris cornell's band?
 - Focus: What
 - LAT: band
 - Q Class: Factoid:band
- how many people were killed in the oklahoma city bombing
 - Focus: How many
 - LAT: many
 - Q Class: Factoid:numerical
- where does the name platte come from?
 - Focus: Where
 - LAT: origin ("come from")
 - Q Class: Factoid:origin
- when did freddie mercury die
 - Focus: When
 - LAT: numerical:date
 - Q Class: Factoid:numerical

1.3 Information Extraction (Knowledge Base Population)

1.3.1 Introduction:

- IE has 2 main components, **named entity recognition (NER)** and **relation extraction (RE)**.
- NER is about **identifying mentions** of names in a text passage and **labeling** them with a **entity type**
 - to populate a KB, NEs need to be **normalized** (aka canonicalized)
 - * Entity normalization involves **disambiguation** of entities as well: e.g. JFK may represent the president JFK or an airport or a museum
 - another task is to **identify mentions** in the text and **link** it with **normalized entity** names in a KB called **Entity Linking**
- **Relation extraction** involves **identifying a relation** between 1 or more entities
 - Relations also need to be **normalized** (aka canonicalized) before adding them to a KB
- <https://web.stanford.edu/~jurafsky/slp3/21.pdf>

1.3.2 NER

1. Introduction

- There are several ways of doing NER
 - Rule based using Regular Expressions or Word shapes
 - From a Gazette
 - Statistical (using sequence models like MEMMs or RNNs)
- Features
 - Syntactic: POS, Dependency Parse, Constituency Parse
 - Lexical:
 - * Words
 - * Word shapes and shorted word shapes
 - e.g. Xanax can be captured by Xxxxx or Xx (shortened form)
- In commercial products, done in a pipeline of 4 stages
 - (a) Use high precision rules to identify unambiguous entity mentions (low recall)
 - (b) Match substrings of NEs found in step 1:
 - (c) Use Gazette or list of names
 - (d) Statistical: high recall low precision
- The reason for stages approach is that:
 - First, some of the entity mentions in a text will be more clearly indicative of a given entity's class than others.
 - Second, once an unambiguous entity mention is introduced into a text, it is likely that subsequent shortened versions will refer to the same entity

2. References

- Coursera NER Slides
- cs224n slides

3. Entity Linking & Disambiguation

- Entity Linking meets Word Sense Disambiguation: a Unified Approach - Moro et. al - TACL 2014 - c202]

- Learning to link with wikipedia - Milne & Witten - CIKM 2008 - c1055
- Robust Disambiguation of Named Entities in Text - Hoffart et al. - EMNLP 2011 - c450
- Collective annotation of Wikipedia entities in web text - Kulkarni et al - KDD 2009 - c364
- USTC submission for TAC KBP EDL 2016
- York Univ submission for TAC KBP EDL 2016
- Entity Linking at Web Scale - WashU/OpenIE - 2012 - c71

1.3.3 Relation Extraction (RE)

1. Introduction

- **Relation:** consists of a set of ordered tuples over elements of a domain, where the domain elements correspond to named entities. For example in the figure below, the relation 'PartOf' consists of tuples $\langle a, b \rangle$ and $\langle c, d \rangle$ which means a is part of b and c is part of d, e.g. 'united airlines' is part of UAL and 'american airlines' is part of AMR.

Domain	$\mathcal{D} = \{a, b, c, d, e, f, g, h, i\}$
United, UAL, American Airlines, AMR	a, b, c, d
Tim Wagner	e
Chicago, Dallas, Denver, and San Francisco	f, g, h, i
Classes	
United, UAL, American, and AMR are organizations	$Org = \{a, b, c, d\}$
Tim Wagner is a person	$Pers = \{e\}$
Chicago, Dallas, Denver, and San Francisco are places	$Loc = \{f, g, h, i\}$
Relations	
United is a unit of UAL	$PartOf = \{\langle a, b \rangle, \langle c, d \rangle\}$
American is a unit of AMR	
Tim Wagner works for American Airlines	$OrgAff = \{\langle c, e \rangle\}$
United serves Chicago, Dallas, Denver, and San Francisco	$Serves = \{\langle a, f \rangle, \langle a, g \rangle, \langle a, h \rangle, \langle a, i \rangle\}$

Figure 21.10 A model-based view of the relations and entities in our sample text.

- **RDF** or resource description format used for storing facts in DBpedia as a relation-entity-relation tuple
- **Ontology (is-a/hypernym):**
- **References:**
 - A Survey of Deep Learning Methods for Relation Extraction - Shantanu Kumar - 2017
 - A Review of Relational Machine Learning for Knowledge Graphs - Nickel et al (MIT) - IEEE - 2016
 - Stanford cs224u (NLU) slides (very nice overview)
 - * Local Copy
 - coursera RE slides
 - Survey Paper - 2008 - 317c (seems outdated)
 - Advances in Automated Knowledge Base Construction - 2013

2. Rule Based RE

- Uses Lexico-Syntactic Patterns, e.g.
 NP_0 such as $NP_1\{NP_2 \dots, (and|or)NP_i\}, i \geq 1$
implies the following semantics:
 - $NP_i, i \geq 1, \text{hyponym}(NP_i, NP_0)$

The above rule allow us to infer the following fact:

- hyponym(Gelidium,red algae) from the sentence "Agar is a substance prepared from a mixture of red algae, such as Gelidium, for laboratory or industrial use."

- High Precision but low recall

3. Supervised IE

- Relation Classification via Multi-Level Attention CNNs - Wang et al - ACL 2016

4. Bootstrapping

- **Introduction:** Start with a seed set of tuples to bootstrap the process of discovering new lexico-syntactic patterns for expressing a relation
 - Bootstrapping proceeds by taking the entities in the seed pair, and then finding sentences (on the web, or whatever dataset we are using) that contain both entities. From all such sentences, we extract and generalize the context around the entities to learn new patterns. Fig. 21.14 sketches a basic algorithm.
- **Algorithm:**
 - Input: A seed set of tuples T, empty set P
 - Repeat:
 - * Find every sentence with mentions $\langle m1, m2 \rangle$ s.t. $\langle m1, m2 \rangle$ corresponds to a tuple
 - Extract pattern p from the sentence between and around m1 and m2
 - * If high confidence in p then add it to a set of patterns P
 - * For each new pattern p (added in the previous step) in P, find all sentences matching p with mentions $t = \langle m1, m2 \rangle$.
 - If $\langle m1, m2 \rangle$ has high confidence, add t to T
- **Semantic Drift:** In semantic drift, an erroneous pattern leads to the introduction of erroneous tuples, which, in turn, lead to the creation of problematic patterns and the meaning of the extracted relations ‘drifts’.
 - Consider the following example:
 - * Sydney has a ferry hub at Circular Quay.
 - * If accepted as a positive example, this expression could lead to the incorrect interest of the tuple Sydney,CircularQuay. Patterns based on this tuple could propagate further errors into the database.
- **Confidence in a Pattern:**
 - hits: the set of tuples in T that p matches while looking in D
 - finds: The total set of tuples that pattern p finds in D
 - The following equation computes the confidence in a pattern p

$$Conf(p) = \frac{hits_p}{finds_p} \times \log(finds_p)$$

- **Confidence in a tuple:** If a tuple t matches P’ patterns, then its confidence is computed as:

$$Conf(t) = 1 - \prod_{p \in P'} (1 - Conf(p))$$

5. Distant Supervision

- Combines the advantages of Supervision and Bootstrapping
- Use entity tuples found in a KB for a given relation and use them for as an indirect mechanism for supervision where the tuples are the input and their corresponding relation in the KB is the target

- **Algorithm:**

- Input: Knowledge Base D, Text T
- for each relation R:
 - * for each tuple $\langle e1, e2 \rangle$ with relation R in D:
 - S \leftarrow sentences in T that contain e1 and e2
 - f \leftarrow frequent features in S
 - O \leftarrow O + new training tuple $\langle e1, e2, f, R \rangle$
- C \leftarrow Train supervised classifier on O
- return C

- Only works for relations for which a large enough database already exists
- Distant supervision for RE without labeled data - 2009 - 1019c
- Learning Syntactic Patterns for Automatic Hypernym Discovery- 2004-585c
- Toward an Architecture for Never-Ending Language Learning - AAAI 2010 - CMU - c928
 - Coupled semi-supervised learning for information extraction - WSDM 2010 - CMU - c303

(a) NN Models for distant supervised RE

- Jointly Extracting Relations with Class Ties via Effective Deep Ranking - Ye et al - 2016
- Incorporating Relation Paths in Neural Relation Extraction - Zeng et al - 2016

6. Unsupervised RE

- **Reverb**

- Run a POS tagger and entity chunker over every sentence in a Corpus S
- Match a regex pattern of POS tags on each sentence s in S (e.g. V|VP|VW*P)
 - * find longest sequence of matching words p in s (merging consecutive matches)
- Check that p satisfies lexical constraint, i.e. it occurs in a minimum # of unique tuples in S
- Find nearest noun phrase e1 to the left and e2 to the right of p (ensuring that the NPs are not relative pronoun or wh-word)
- Assign confidence c to the relation $r = (e1, p, e2)$ using a confidence classifier

(a) WashU (openIE papers)

- Open Language Learning for Information Extraction (OLLIE) - 2012 - 293c
- Open Information Extraction: The Second Generation - 2011 - 321c
- Identifying Relations for Open IE (REVERB)- 2011 - 708c
- Semantic Role Labeling for Open IE (SRLIE)- 2011 - c69
- Open Information Extraction from the Web (TextRunner) - 2007 - 1706c
- Web-scale information extraction in knowitall - 2004 - 958c
- Open Information Extraction Using Wikipedia - 2010 - 379c
- <http://openie.allenai.org/>
- The most important papers are (in order of importance)
 - OLLIE
 - REVERB
 - Open IE: The Second Generation
 - SRLIE
 - TextRunner

7. RE using Matrix Factorization or Distributed Representations

- Start with a sparse matrix with rows as entity tuples and columns as relations and each entry as a binary value (1 implies that the tuple and relation form an observed fact and 0 means the corresponding fact is unobserved)
 - Here relations could be a set of words (occurring with sufficient frequency e.g. at-least with 10 distinct pairs of tuples) representing a predicate v in the shortest $(s \rightarrow v \leftarrow o)$ path in a dependency parse or a normalized relation in a KB
 - Using matrix factorization, we discover low order manifolds of the entity-tuple as well as relation space, i.e. low rank representations of the entity vectors as well as the relation vectors
 - Logistic function on the dot-product between a tuple vector and a relation vector allows us to predict whether the combination of the two form a fact.
 - It also allows us to find similar relations
 - It also allows us to predict implicature, i.e. historian-at relation implies professor-at relation
 - Hybrid of Unsupervised learning and distant supervision
- Modeling Relations and Their Mentions without Labeled Text - Reidel et. al - 2010 - c275
- Relation Extraction with Matrix Factorization and Universal Schemas - Reidel - 2013 - HLT NAACL - c256
 - Local Annotated Copy
- Translating Embeddings for Modeling Multi-relational Data - Weston et al - NIPS 2013 - c296
- Knowledge Graph Completion via Complex Tensor Factorization - Riedel et al - 2017

8. Canonicalizing Relations

- Unsupervised Relation Discovery with Sense Disambiguation - Yao, Reidel, McCallum- ACL 2012
- DIRT – Discovery of Inference Rules from Text - Lin & Pantel - 2010 - c471

9. Other Important Papers

- Knowledge vault: a web-scale approach to probabilistic knowledge fusion - Google - KDD 2014 - c357
- DeepDive: Web-scale Knowledge-base Construction using Statistical Learning and Inference - WisMed - VLDS 2012 - c72

1.4 QA from Structured KB

1.4.1 KBs

- DBpedia - A crystallization point for the Web of Data - Bizer et al - 2009 - c1442

1.4.2 KBQA

1. Mapping Questions to a KB Query by Paraphrase Learning (Wash U)

- Open question answering over curated and extracted knowledge bases - 2014

2. PARALEX

- **Setting:**
 - The knowledge source is a KB, i.e. a DB of entities and relations (with formal identifiers).
 - We use the following DB concepts:
 - * Question Templates, e.g. Who is the r of e ?
 - * Entities
 - * Relations

- Our goal is to find mappings from question sub-strings

p

to objects in the DB

d

. A mapping is denoted

(p, d)

• Mapping Question to a KB Query

- First a lexicon is learnt which allows mapping disjoint sub-strings (ie. sub-phrases) of the question to a question template, an entity and a relation in a KB.
- The mapping proceeds in 2 steps. First a question is mapped to a set of question templates with gaps for relation and entity. In the second step, the gaps for relation and entity are filled in. Note that a single string may map to several entities or relations.
- The lexicon is learnt using question paraphrases where each question can be expressed in several different ways (i.e. question cluster)
 - * Who wrote the Winnie the Pooh books?
 - * Who is the author of winnie the pooh?
 - * What was the name of the authur of winnie the pooh?
 - * Who wrote the series of books for Winnie the poo?
 - * Who wrote the children's storybook 'Winnie the Pooh'?
 - * Who is poohs creator?
- Paraphrases allow you to learn multiple ways of expressing the same Q-template (or entity or relation)
- Also observe that a word/phrase may map to several different entities/relations/templates because the same word/phrase may mean different things in different contexts (here context is a Q cluster)
 - * e.g. radiation -> radiation, electromagnetic-radiation, nuclear-radiation
- This means that a question can map to several different queries. To prune the queries, we use ML model to rank the derived queries. Here, each mapping

(p, d)

is treated as a feature and a parameter is learnt for it.

• Lexicon Learning:

- The lexicon is learnt using question paraphrases (from WikiAnswers) and a seed lexicon
 - * **Seed Lexicon:** consists of 16 question templates and relation and entity strings of relation/entity identifiers in REVERB DB.
 - e.g. entity id "new-york" in REVERB maps to entity string "new york" and relation id "treatment-for" maps to relation string "treatment for"
 - * Roughly, the algorithm first checks if a question x (in a paraphrase cluster) can be parsed just using the seed lexicon (i.e. results in mappings from words/phrases to template/entity/relations). If true, then using a word alignment algorithm, we first induce a word by word alignment

A

b/w a paraphrase x' of x and x . Using word-alignment

A

, we create phrase mappings from a phrase in x' to a template/entity/relation in the seed lexicon. This allows us learn new mappings from phrases to DB concepts and extend our lexicon.

- This is done for all paraphrases x' of x

- **Learn parameters for Ranking Mappings**

- The training set for this is generated using the seed lexicon and the paraphrase corpus itself. See Figure 2 of the document.

- Paraphrase-Driven Learning for Open Question Answering- 2013 - c152 (aka PARALEX)

3. Mapping Questions and Triples to BOW based Vector Embeddings

- Here we simply map each Question to a BOW and then to a Vector embeddings which is a sum of embeddings of across all the words in the question
- Facebook Papers in order of progression in time
 - OpenQA with Weakly Embedded Models
 - QA with subgraph embeddings
 - Memory Networks
 - Large Scale Simple QA

1.4.3 Attention

1. LSTM

- Improved Neural Relation Detection for Knowledge Base Question Answering - IBM research - 2017
 - Local Annotated Copy
 - **Terminology:**
 - * **Entity Linking (EL):** entity linking links n-grams in questions to KB entities
 - **Objective:**
 - * use EL and relation detection (RD) in Qs to search for answers in a KB
 - **Challenges in RD in the context of KBQA:**
 - * typically supervised RE only deals with 100 relations while KBQA on a small KB like Freebase2M deals with 6K relation types
 - * Zero shot learning: relations observed at test time not observed in training data
 - * Relation Chaining: need to chain more than 1 relation to answer Qs in WebQuestions
 - **Data Set:** SimpleQuestions (this set has relations in the test set not covered in the training set) and WebQSP
 - **Introduction:**
 - **Relation Detection at Different Levels of Granularity**

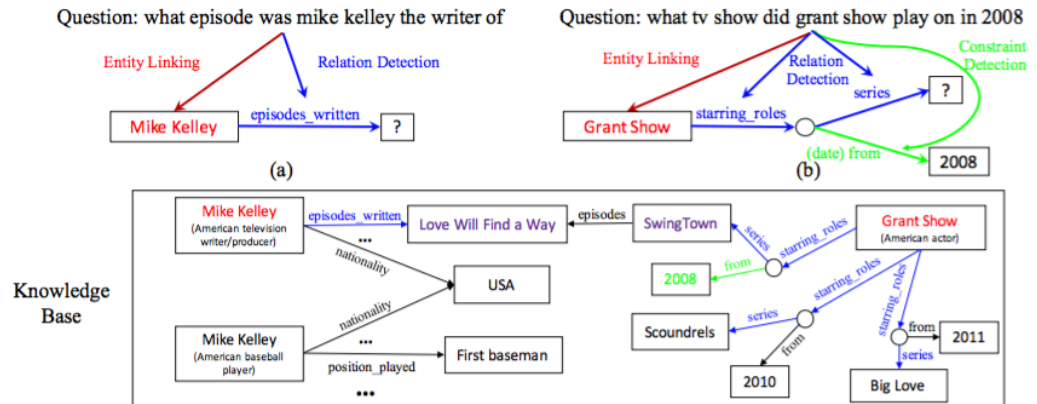


Figure 1: KBQA examples and its three key components. (a) A single relation example. We first identify the topic entity with **entity linking** and then detect the relation asked by the question with **relation detection** (from all relations connecting the topic entity). Based on the detected entity and relation, we form a query to search the KB for the correct answer "Love Will Find a Way". (b) A more complex question containing two entities. By using "Grant Show" as the topic entity, we could detect a chain of relations "starring_roles-series" pointing to the answer. An additional **constraint detection** takes the other entity "2008" as a constraint, to filter the correct answer "SwingTown" from all candidates found by the topic entity and relation.

* Done at **multi-levels** of Relation Representation:

- **Word-level representation:**

- to deal with unseen relations during test time
- better generalization
- the word-level focuses more on local information (words and short phrases) and **not** on global information (which relation-level does)
- For example, in the figure above, when doing only word-level matching, it is difficult to rank the target relation “starring_{roles}” higher compared to the incorrect relation “plays_{produced}”. This is because the incorrect relation contains word “plays”, which is more similar to the question (containing word "play") in the embedding space

- **Relation-Level Representations:**

- Original relation names can sometimes help to match longer question contexts (**global information**) but suffers from data-sparsity problems.
- relation-level representation matches long phrases and skip-grams from the question
- here **relation-name** is used as a unique token
- **data-sparsity problem:** low coverage of relations as number of relation types in a KB are limited
- e.g. difficult to match the questions to relation names “episodes_{written}” and “starring_{roles}” if these names do not appear in training data
- Example Q: "what tv episodes were <e> the writer of"
- At the relation-level, the longer phrase "tv episodes were <e> the writer of" matches relation name "episodes_{written}"
- On the hand, at word-level, short phrases like "tv episodes" match the word "episodes" and "the writer of" matches the word "written" in the "episodes_{written}" relation name.

– **Improved Relation Detection Model:**

* **Relations Representations at Different Granularities:**

- 2 Bi-LSTMs:
- one which represents relations as words
- e.g. for a relation chain containing relations, "starring_{roles}" and "series", the input to this LSTM are the words {starring, roles, series}
- one which represents relations using relation names
- e.g. for a relation chain containing relations, "starring_{roles}" and "series", the input to this LSTM are the relations {starring-roles, series}

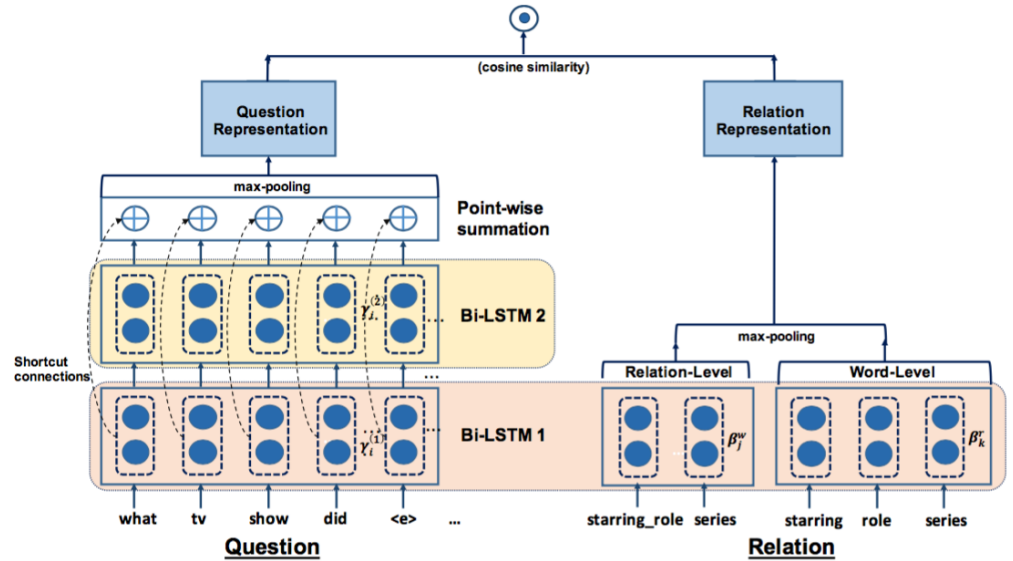
* **Question Representations:**

- **Deep Bi-LSTM:** comprised of 2 layers of Bi-LSTM where the input to the second layer are the hidden vectors outputted by the first layer.
- Allows question representations comprised of vectors that summarize phrase information of various lengths (in order to match relation representations of different granularity)
- Since the second Bi-LSTM starts with the hidden vectors from the first layer, intuitively it could learn more general and abstract information compared to the first layer.

* **Residual (Hierarchical) Matching b/w Relation & Question**

- here we expect that two layers of question representations can be complementary to each other and both should be comparable to the relation representation space
- This is important for our task since each relation token can correspond to phrases of different lengths, mainly because of syntactic variations

- For example the relation word "written" could be matched to either the same single word in the question or a much longer phrase "be the writer of" depending on whether the question is "what episode was written by <e>" or "what tv episodes were <e> the writer of" respectively.
- **Weighted Sum Model and its problems:** We could perform the above hierarchical matching by computing the similarity between each layer of LSTM output and the relation representation separately and doing the (weighted) sum between the two scores.
- **Weighted Sum Model** model performs much worse (even in training) compared to residual model.
- Reason for this are:
 - Deep BiLSTMs do not guarantee that the two-levels of question hidden representations are comparable, the training usually falls to local optima where one layer has good matching scores and the other always has weight close to 0
 - To overcome the above difficulties, we adopt the idea from Residual Networks (He et al., 2016) for hierarchical matching by adding shortcut connections between two BiLSTM layers. See figure Below.



Here we do point-wise sum of output from cells of each of the 2 Bi-LSTM layers

- Intuitively, the proposed method should benefit from hierarchical training since the second layer is fitting the residues from the first layer of matching, so the two layers of representations are more likely to be complementary to each other. This also ensures the vector spaces of two layers are comparable and makes the second-layer training easier.

* **KBQA enhanced by Improved Relation Detection:** Consists of following steps:

- **Entity Linking:** this paper uses top-K entities linked to the Q using an existing entity linker. Call these

$$EL_K(q)$$

- **Entity Reranking:** Use the raw question text as input for a relation detector (e.g. HR-BiLSTM) to score all relations in the KB that are associated to the entities in

$$EL_K(q)$$

; use the relation scores to rerank

$$EL_K(q)$$

and select the top-K' entities based on the reranking. Denote these entities as

$$EL_{K'}(q)$$

- because the original entity linking cannot distinguish between "Mike Kelley" the TV writer with "Mike Kelly" the baseball player.
- **Exploiting Entity-Relation Collocation in the Q** entities returned by entity linking are reranked based on how they are linked to relations (in the KB) which are associated with i.e. similar to those in the question (in the embedding space). This helps with disambiguation of entities returned by the original entity-linker
- **Rerank Score:** a score produced by entity reranking step for each entity

$$e \in EL_K(q)$$

is produced as follows:

$$s_{rerank}(e; q) = \alpha \cdot s_{linker}(e; q) + (1 - \alpha) \cdot \max_{r \in R_q^l \cap R_e} s_{rel}(r; q)$$

·

$$R_e$$

: set of relations associated with entity

$$e$$

·

$$s_{rel}(r; q)$$

: score generated by a relation detection model (like HR-BiLSTM) when matching relation

$$r$$

with question

$$q$$

·

$$R_q^l$$

: set of

$$l$$

best scoring relations associated with entities in

$$EL_K(q)$$

·

$$s_{linker}(e; q)$$

: score generated by the entity linking component for linking entity

$$e$$

in KB to question

$$q$$

- Based on the rerank score, top-K' entities

$$EL_{K'}(q)$$

are selected

- **Relation Detection:** Detect relation(s) using the reformatted question text in which the topic entity is replaced by a special token <e>. This helps the model better distinguish the relative position of each word compared to the entity.

- This step produces a score

$$s_{rel}(r; e, q)$$

for each entity

$$e \in EL_{K'}(q)$$

- **Query Generation:** outputs the <entity, relation (or core-chain)> pair

$$s(\hat{e}, \hat{r}; q) = \max_{e \in EL_{K'}(q), r \in R_e} (\beta \cdot s_{rerank}(e; q) + (1 - \beta) \cdot s_{rel}(r; e, q))$$

- **Constraint Detection:** Consists of 2 steps:
- **Sub-graph generation:** given the top scored query generated by the previous 3 steps, for each node v (answer node), we collect all the nodes

$$c$$

connecting to v (with relation

$$r_c$$

) with any relation, and generate a sub-graph associated to the original query

- **Entity-linking on sub-graph nodes:** we compute a matching score between each n-gram in the input question (without overlapping the topic entity) and entity name of

$$c$$

(except for the node in the original query) by taking into account the maximum overlapping sequence of characters between them . If the matching score is larger than a threshold (tuned on training set), we will add the constraint entity

$$c$$

(and

$$r_c$$

) to the query by attaching it to the corresponding node

$$v$$

on the core-chain.

- Dataset and Neural Recurrent Sequence Labeling Model for Open-Domain Factoid Question Answering - Baidu - 2016
- Question Answering on Freebase via Relation Extraction and Textual Evidence - ICST Peking University - ACL 2016
-

2. Using Key-Value MemNN

- Key-Value Memory Networks

3. Using CNN

- Simple Question Answering by Attentive Convolutional Neural Network - Yin & Schutze - 2016
- State of Art

1.4.4 Inference in KBs

- Random Walk Inference and Learning in A Large Scale Knowledge Base - CMU - EMNLP 2010 - c169
- Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks - Das et al. (UMass) - 2016

1.5 IR based QA

1.5.1 Semantic Search

- Placing search in context: the concept revisited - Finkelstein et al - 2002 - c890
 - local copy
- Semantic search - Guha et al. 2003 - c622

1.5.2 Query Preprocessing

1. Answer Type detection

- Learning Question Classifiers - Li, Roth - 2002 - c543
 - Annotated Copy
- Learning question classifiers: the role of semantic information - Li, Roth - 2006 - c129
- Question analysis: How Watson reads a clue - IBM - 2012 - c95

2. Question Type Classification

- Skip-Thought Vectors - Kiros et al - 2015 - c97
 - https://github.com/tensorflow/models/tree/master/skip_thoughts

1.5.3 Query Formulation

- LASSO: A Tool for Surfing the Answer Net - Maldovan et al - 1999 - c168
- The Structure and Performance of an Open-Domain Question Answering System - Maldovan et al - acl 2000 - c104

1.5.4 Important Papers

1.5.5 DataSets

1.5.6 Overall Pipeline* NLP Coursera Slides

1.6 Answer Sentence Selection

1.6.1 LDC Method

- Sentence Similarity Learning by Lexical Decomposition and Composition - IBM research - COLING 2016
 - local copy
- **Goal:** Given a set of sentences relevant to a Question, select the sentence that best answers a question
- **Paper Overview:**

- Uses a CNN model with max-pooling to measure similarity between questions and answers. It specifically addresses 3 issues in comparing similarity between question and answer:
 - * Use of different lexicons (words)
 - This is addressed using word vectors
 - * Similarity comparison at various levels of granularity, i.e. word-level, phrase level, syntax level
 - This is addressed in 2 ways:
 - For each word in the Q find the closest word(s) in the sentence and vice-versa
 - Use CNN filters of varying size to capture similarities at unigram, bigram and trigram levels
 - * Dissimilarity between Q and Sentence is also a significant clue
 - The word vector for each word in the Q is decomposed into similar and dissimilar components (by projecting it onto a weighted sum of most similar words in the Sentence). The same is done for each word in the sentence)

- **Data:**

- WikiQA: The knowledge source here is Wikipedia and the Questions are created based on real Bing Queries. There are approximately 2.3K questions in this corpus (2K for training, 100 for validation and 250 for test)

- **Results on our Implementation:**

- We achieved the result of 73% MRR (mean reciprocal rank) and 70% MAP (mean average precision) as specified by the paper and what is the state of the art.

1.6.2 Important Papers

- Deep Learning for Answer Sentence Selection - Deepmind - 2014 - c101
 - local annotated copy

1.7 Machine Comprehension

1.7.1 multi-context using seq2seq model

- Teaching Machines to Read and Comprehend - 2015 - 294c
- Text Understanding with the Attention Sum Reader Network - Kadlec et al - c51 - 2016
- SQuAD: 100,000+ Questions for Machine Comprehension of Text - Stanford - 2016
- Reading Wikipedia to Answer Open-Domain Questions - Facebook AI - 2017
 - local annotated copy
- A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task - Stanford - 2016
 - Annotated Copy
- R-NET: MACHINE READING COMP - Microsoft Research - 2017
- Recurrent Entity Networks

1.7.2 Using Memory Networks (from of attention)

- Key-Value Memory Networks

1.7.3 Using Manually Curated Syntactic/Semantic Features

- Learning surface text patterns for a Question Answering System - Ravichandran et al - 2002 - c778
- Modeling biological processes for reading comprehension - Stanford - 2014 - c77
- Learning answer-entailing structures for machine comprehension - Sachan et al - 2015
- Machine comprehension with syntax, frames, and semantics - Wang et al - 2015

1.7.4 DataSets

- MCTest: A Challenge Dataset for the Open-Domain Machine Comp of Text - 2013 - c139
 - a set of 660 stories and associated questions intended for research on the machine comprehension of text.

1.8 Memory Networks:

1.8.1 OpenQA with Weakly Embedded Models

- open QA with weakly embedded models
- **Objective:** Open QA on an auto-generated KB known as the ReVerb KB
 - The main idea is to score a question against all the triples in the KB and return the best scoring triple
- **Vector Embedding:** Introduces the idea of using vector embeddings for words to do QA
- **BOW vector:** Uses the idea of BOW vector embedding for a sentence, i.e. a sentence is considered as a BOW (with word order not mattering). So the vector embeddings for all words in a sentence are added up to form a vector for the sentence
- **Scoring:** $s(q, t) = f(q)^T g(t)$

Where $f(q) = V\Phi(q)$ and $g(t) = W\Psi(t)$

Where V & W are embedding matrices, while $\Phi(q)$ and $\Psi(t)$ are BOW vector encoding presence/absence of dictionary words in a question q and triple t
- **Fine Tuning:** a matrix $M \in R^{k \times k}$ parameterizing the similarity between words and triples embeddings is used:
$$s_{ft}(q, t) = f(q)^T M g(t)$$
- **String Matching:** To avoid scoring all triples (consequently reducing noisy matches), only triples which have at least one word matching the question are considered
- **Dataset:**
 - ReVerb KB
 - Question Paraphrases from WikiAnswers
 - Auto-generated QA from ReVerb KB
 - WebQuestions

- **Implementation Details**

- Questions generated from ReVerb triples used templates (with some constraints, i.e. some relation types can't be used with some templates). The entity and relationship tokens were converted into words when used inside the question. replacing - by spaces and stripping of their suffixes, e.g the string representing 'winston-churchill.e' is simply 'winston churchill'.
- * The answer for the generated question is the triple itself from which the question was generated.
- model ends up learning vector embeddings of symbols, either for entities or relationships from ReVerb (present in the answer triple for the auto-generated QA), or for each word of the vocabulary.
- An entity doesn't have the same embedding when it appears on the left vs on the right in a triple. So 2 different embedding dictionaries are used. This is to reflect the fact that a relationships in the KB are not symmetric
- Representing questions as a bag of words might seem too limited, however, in this particular setup, syntax is generally simple, and hence quite uninformative. A question is typically formed by an interrogative pronoun, a reference to a relationship and another one to an entity. Besides, since lexicons of relationships and entities are rather disjoint, even a bag of words representation should lead to decent performance, up to lexical variability. There are counter-examples such as What are cats afraid of ? vs. What are afraid of cats ? which require different answers, but such cases are rather rare. Future work could consider adding parse tree features or semantic role labels as input to the embedding model
- Use margin loss with the constraint that embeddings in V and W have a norm of ≤ 1 .

$$f(q_i)^T g(t_i) > 0.1 + f(q_i)^T g(t') \quad \forall i, \forall t' \neq t_i$$
and $\forall i, ||v_i||_2 \leq 1$ and $\forall j, ||w_j||_2 \leq 1$
- Given a (q, t) , create a corrupted triple t' with the following method: pick another random triple t_{tmp} from the KB, and then, replace with 66% chance each member of t (left entity, relationship and right entity) by the corresponding element in t_{tmp} . This heuristic creates negative triples t' somewhat similar to their positive counterpart t
- Embeddings are initialized s.t. each word vector has a mean of 0 and std. deviation of $1/k$ where k is the length of the vector.

1.8.2 QA with subgraph embeddings

- QA with subgraph embeddings - 2014 - 159c
- **Key Idea:** This model supports Q&A over a human curated KB (Freebase in this paper). Each question is supposed to reference an entity in the KB. All entities which are connected to the entity in the question by 1 or 2 hop paths are considered as candidate answers (i.e. potential answers). Each candidate answer is mapped to a vector embedding using the subgraph of entities connected to it (as explained below). This vector is scored against the embedding vector of the question (using dot product) and the highest scoring candidate is predicted as the answer.
- **Subgraph Representation** of a candidate answer is done by summing the embedding vectors of the following:
 - All the entities directly connected to the candidate answer (along with their relation to the candidate)
 - The candidate itself
 - The entity in the question and the relations along the 1 or 2 hop path to the candidate

- **Identifying KB entity in the Question** when the questions are generated from the KB, then the entity is explicitly identified. Otherwise some form of string matching b/w the words in the question and the entity MIDs in the KB is used to find the KB entity.

- **Note:** that each entity in Freebase KB is identified by an MID.

- **Dataset:**

- WebQuestions : corpus generated using Google Suggest API where the Question & Answers reference Freebase entities
- Auto-generated Qs from Freebase
 - * Freebase is organized as triplets (subject, type1.type2.predicate, object), where two entities **subject** and **object** (identified by mids)
 - * We used a subset, created by only keeping triples where one of the entities was appearing in either the WebQuestions training/validation set or in ClueWeb extractions.
 - * All triples were converted into questions “What is the predicate of the type2 subject?” (using the mid of the subject) with the answer being object.
- Auto-generated Qs using ClueWeb extractions - String matching is done b/w ClueWeb Extractions words and Freebase Entities to generate 2M extractions structured as (**subject**, “text string”, **object**) with both **subject** and **object** linked to Freebase. We also converted these triples into questions by using simple patterns and Freebase types
- WikiAnswer’s Question Paraphrases

- **Implementation**

- **Margin based Ranking Loss**
- **Negative candidate selection:** This is achieved by sampling 50% of the time from the set of entities connected to the entity of the question (i.e. other candidate paths), and by replacing the answer entity by a random one the other 50% of times.
- Embedding vector norms constrained to be ≤ 1
- **Scoring MID against the actual words in the MID name:** Training of embedding vectors is multitasked with the mapping of the Freebase MIDs to the actual words of their names
- **Candidate Answer set reduction:** instead of using the all the triples in the KB as the candidate set, we only use triples which contain the entity referenced in the question (i.e. 1 hop away) or all entities which are 2 hops away from the question entity (further restricted using beam search to 2-hop entities connected via 10 first hop relations most likely to answer the question. All first-hop relations from the question entity are scored against the question and only the top 10 are kept).
- **Multiple Answers:** A question can have multiple answers, i.e. a set of entities. For example for questions like “Who are David Beckham’s children?”. All entities that lie on the same 1 or 2-hop path from the question entity are considered as a candidate set of answers. The feature representation of the prediction is then the average over each answer’s features (Here by features we mean how the embedding vector of the candidate answer entity is comprised using the subgraph representation).

- **Results**

- (on WebQuestions):
 - * Subgraph & A(q) = C2: P@1 = 40.4, F1(Berant) = 39.2, F1(yang) = 43.2
 - * Ensemble with (Berant & Liang, 14): F1(Berant) = 41.8 F1(Yang) = 45.7

1.8.3 Memory Networks

- Memory Networks
- **Dataset:**
 - A subset of BabI (for Simulated World QA task)
 - ReVerb + WikiAnswers Dataset (for large scale simple QA task)
 - * ReVerb KB
 - * Question Paraphrases from WikiAnswers
 - * Auto-generated QA from ReVerb KB

1.8.4 Towards AI-Complete QA

- Towards AI-Complete QA
- **Dataset:** BabI (introduced by this paper)

1.8.5 End-to-End Memory Networks

- end-to-end memory networks
- **Key Idea:** Memory networks required separate training for each hop because they were using a hard max when comparing Q to memories. This also required strong supervision, i.e. a supporting sentence for each intermediate hop. To get around this, end-to-end memory networks were invented. They replace the hard max with a soft max allowing back-propagation across multiple hops.
- **Dataset:** BabI

1.8.6 Large Scale Simple QA

- Large-scale Simple Question Answering with Memory Networks - 2015 - c70
 - local annotated copy
- **Datasets:**
 - SimpleQuestions (Introduced by this paper)
 - * generated by Mechanical Turks based on **Freebase Tuples**
 - WebQuestions
 - Auto-generated Questions from Freebase
 - Question Paraphrases from WikiAnswers
 - Gold set of QA on Reverb Set (introduced by Paralex paper)
- **Transfer Learning:** learning done on Simple Questions and Auto-generated Freebase Questions can be applied to ReVerb without training
- **Removing mediator nodes:** in freebase reduces path length between entities from 2 to 1, widely increasing the scope of simple QA task on Freebase.

- **Grouping objects** of (subject, relation, object) relations with common subject and relation values
- **Feature Representation of facts:**
 - Freebase Facts: using Bag-of-symbol vector where symbols are entities or relationships in Freebase
 - Questions: using Bag-of-ngrams vector (the dictionary consists of individual words appearing in Questions as well as aliases of Freebase Entities, each alias being a single n-gram)
 - Reverb Facts:
 - * relations encoded using bag-of-words
 - * entities encoded using either Freebase symbol or Freebase alias or bag of words.
- **Candidate Generation:** to avoid matching all facts in the KB with the question, a set of candidate facts are generated which are then used for matching. The candidate sets are generated by generating all possible n-grams of the question (which do not contain an interrogative noun or a stop word). Only n-grams which are an alias of a Freebase entity are kept. All n-grams which are sub-sequence of an n-gram which is kept are discarded. For 5 longest matching n-grams, 2 entities with most links in Freebase are kept. All facts with these entities are candidates for matching.
- supercedes Question Answering with Subgraph Embedding
- **Results:** (in F1 units)
 - WebQuestions = 41.9, SimpleQuestions = 63.9, Paraphrases = 68

1.8.7 Key-Value Memory Networks

- Key-Value Memory Networks for Directly Reading Documents - EMNLP - 2016
 - local annotated copy
- **Overall Paper**
 - Compares QA performance across 3 knowledge source types for the same knowledge (movies in this case), i.e.
 - * human annotated KB (OMDB/wikilens) - maps to **QA from Structured KB**
 - * IE based KB (extracted using IE techniques from Wikipedia articles of movies in OMDB) - maps to **QA from Structured KB**
 - * Document (Wikipedia) - maps to **Machine Comprehension** described below in the section on 1
 - Data
 - * (WikiMovies): 120k QA pairs programatically generated generated using templates derived from SimpleQuestions which are then applied to OMDB/Wikilens based high quality QA (split into ~100K training, 10K Dev and 10K test set).
 - * WikiQA: 1000 training and 1000 test questions based on random wiki articles
 - Results: (%hits@1)

* WikiMovies: KB = 93.9, IE = 68.3, Doc = 76.2

* WikiQA: MAP = 0.7069, 0.7265

1. Machine Comprehension: Factoid Q&A using Wikipedia

- **Goal:** The goal here was to answer a factoid question about Movies using Wikipedia. The answer is a single entity, e.g. an actor name or a movie name

- **Main Ideas:**

- Uses the abstraction of a soft memory controller where addressing is content based (i.e. key based) and the memory is read using soft attention (value read from the memory is a weighted combination of memory entries weighted by the how relevant the key is to the question, i.e. you focus attention on those memory entries which are relevant to the question)
- Also ideas like title of an article has a high probability of being an answer is taken from IBM Watson
- Executing Multiple hops of memory access (by updating the question with the value read from the memory in the last hop) is similar to walking a entity relation graph to find an entity multiple hops away from the entity and relation specified in the question

- **Overview from Paper:**

- For each question, apply IR techniques (e.g. inverted index without low-idf/stop words) to select a set of passages from Wikipedia Articles relevant to that question
 - * Here the passages are divided into sequential windows (of 7 words each) with an entity at its center
 - * the window is the key and the entity at the center is the value
- **Key Addressing:** The question is compared with each of the windows (i.e. the key) to compute a relevance score for each window wrt the question. This is essentially a soft attention mechanism to find the keys which are most similar to the question.
- **Value Reading:** values from the memory locations are read by taking their weighted sum using the relevance score.
- This result is combined with the question to formulate a new query which is used to repeat **Key Addressing** and **Value Reading**. These 2 steps are repeated a few times (i.e. # of hops)
 - * The motivation for doing this is that new evidence (i.e. the value read) can be combined with the query to focus and retrieve more pertinent information in subsequent accesses
- The combination of the value read from the last access (hop) and the the last version of the query is compared against every entity (potential answer) in the vocabulary to find the most similar entity which is output as the answer.

- **Data:**

- WikiMovies: 120k QA pairs of programatically generated questions using templates derived from SimpleQuestions (open-domain Q&A dataset based on Freebase posed by human annotaters using crowd-sourcing). Questions pertaining to the Movie genre were identified from the SimpleQuestions dataset and the entities in these were replace by entities in the OMDb/Wikilens based KB. Only those Questions which are answerable using Wikipedia were kept.

- Wikipedia: 17K documents pertaining to movies were selected relevant to the WikiMovies Questions

- **Results on our Implementation:**

- We achieved the result of 76% precision (at rank 1) specified in the paper and what is the state of the art.
- Our implementation was faster than the paper’s own implementation (e.g. our training take around 5 hours while the paper’s own implementation takes a couple of days). This could be because of several factors (use of aggressive negative sampling, GPUs etc). We get convergence in about 40 epochs

- **Observations from Our Implementation:**

- Aggressive negative sampling is quite beneficial. Without it, convergence is slow (150 epochs) and we loose 1% in accuracy (from 76% to 75%)
- Length normalization is also quite helpful. Without it convergence is slow (it takes 114 epochs to get 76% accuracy)
- 2 hops is beneficial compared to 1 hope. The performance loss is 2% (74% with 1 hop and 76% with 2 hops)
- **Some observations on why a question may not get answered**
 - * Missing Coreference resolution
 - * Weakly trained embedding for a specific word (as it may not have encountered often during training) doesn’t affect the result as much as it should
 - * High frequency symbols like u’1:.’ or u’and’ have high norm and seem to add a lot of noise

- **Problems with data**

- Parsing ambiguity in answer string in "train1.txt", e.g. "No Retreat, No Surrender 2" is 1 answer, but we currently assume that a list of answers is a comma separated list
- mismatch in entity name b/w entities.txt and the wiki due to difference in case of a starting letter, e.g. "Anthony ward" in wiki vs "Anthony Ward" in entities.txt. Also 'anti-semitism' in entities.txt is incompatible with 'antisemitism' and 'anti-Semitism' in wiki.txt.
- "Why Did I Get Married Too?" is not recognized as an entity by the parser which uses ' \leq entity-name>' to match an entity but ' \leq ' is not compatible with non-alphanumeric character at the boundary of the entity name

1.8.8 Recurrent Entity Networks

- Tracking the State of the World using REN - Dec 2016

1.8.9 Evaluating Prerequisite Qualities for End-to-End Dialog Systems

- Evaluating Prerequisite Qualities For Learning End-To-End Dialog Systems - 2016

1.8.10 Dialog-based Language Learning

- Dialog-based Language Learning - NIPS 2016

1.8.11 Memory Network Extensions

- **Efficient memory via hashing**
 - See 3.2 of Memory Networks
- **Modelling implicit notion of time encoded in the order of sentences in a story:**
 - **Modelling write time** See 3.4 of Memory Networks
 - **Temporal Encoding:** See 4.1 of end-to-end memory networks
- **Modelling unseen words:** predict a word using its neighbouring (context) words, i.e. for each unseen word, store a bag of words that word has co-occurred with. Use a separate set of embeddings for the context words which is learned at training time by probabilistically pretending that a word has not been seen before. See 3.5 of Memory Networks
- **Exact Matches and Unseen Words:**
 - We would like to improve the matching score if input and memory had some exact word matches.
 - See 3.6 of Memory Networks
- **Predicting multiple words:** use RNN or add a special word w_\emptyset to the dictionary and predict word w_i on each iteration i conditional on previous word i.e.

$$w_i = \operatorname{argmax}_{w \in W} s_R([x, m_{o_1}, \dots, m_{|o|}, w_1, \dots, w_{i-1}], w_k)$$

- for handling multi-word outputs
 - See end of 3.1 in Memory Networks
 - See A.2 of Towards AI-Complete QA
- **Adaptive Memory:** for handling questions requiring chaining of multiple facts
- **Handling Word Order in a Sentence:**
 - **N-grams BOW:** (instead of 1-gram BOW)
 - * For handling cases where word ordering matters e.g. subject vs object
 - * See A.2 of Towards AI-Complete QA
 - **Positional Encoding:** See 4.1 of end-to-end memory networks
- **Non-Linear matching function:** using a matching fn which is a classical 2 layer neural-net with tanh neuron.

$$s(x, y) = E(x)^T E(y)$$

Where,

$$E(x) = \tanh(W \tanh(U \Phi_x(x)))$$

Here W is $n \times n$ matrix

- important for modeling 3-way interactions, e.g for yes/no questions, negations and indefinite knowledge
- See A.2 of Towards AI-Complete QA

1.8.12 Datasets

Dataset Name	Description	Size	Generation method	Curated/ Noisy
Reverb KB	(subject, relation, object) triples mined from ClueWeb09. 2M entities and 600K relations	14m	Mined using Reverb from ClueWeb09	Noisy
Paraphrase Corpus (WikiAnswers)	(q, q') pairs	35m	Human annotated on WikiAnswers	Noisy
Gold set of QA pairs introduced by Paralex Paper	(question, answer, label) triples	48k	Human anotated using question pairs from paraphrase corpus which are answerable using Reverb KB	Curated
Facebook BabI Dataset	20 tasks with 1K/10K Qs	20K/200K	Simulator	Clean
SimpleQuestions	(q,a) pairs, where q is a written in natural language by human annotators using a fact from Freebase. This fact also provides the answer	100K	Human annotators	Clean
WebQuestions	(q,a) pairs where 'a' is a list of aliases for Freebase entities. Many questions in this dataset expect multiple answers	5.8K	Human anotator	Clean
Automatically generated Qs from ReVerb	Generated using 16 seed Q templates applied to ReVerb facts (with some constraints)	250M	Auto-generated	Clean
Automatically generated Qs from Freebase	Generated using 16 seed Q templates applied to Freebase facts (with some constraints)	10/12M	Auto-generated	Clean

1.9 IBM DeepQA (Watson)

1. Summary of the DeepQA Pipeline

- **Introduction:**

- The pipeline consists of 4 stages:
 - * QA Analysis
 - * Candidate Answer Generation
 - * Evidence Gathering & Scoring
 - * Merging Scores & Ranking Candidate Answers
- The stages of Candidate Answer Generation and Evidence Gathering are implemented as ensembles of a large # of components.
- The concept of a dependency parse tree using Deep Parsing is very fundamental to DeepQA as it is used in many components in all the stages of the pipeline

- **Deep Parsing used pervasively as a mechanism:** essentially returns a dependency parse tree with some tweaks which they refer to as PAS (predicate argument structure).
 - The most important property of a PAS tree is that it allows capturing a large variety of expressions of the same logical proposition into a single common structure.
 - * This in turn allows very efficient writing of rules for relation extraction (e.g. one rule suffices for 15 different ways of expressing the authorOf relation)
 - All sentences in the information corpus as well as all questions are parsed using this mechanism. The resulting parse tree is used in a very large # of places from candidate generation to type checking and for many more functions as described below.
- **QA Analysis:** this stage essentially generates a focus, a LAT and a Question type.
 - The focus is a phrase/word in the Question which tells what the question is asking. It can be substituted by a candidate answer and the Question becomes an assertion which make sense as a unit.
 - LAT identifies the type of answer the question is seeking
 - Question Type: determines what kind of a question is being posed (trivia question, puzzle, limerick etc.). This determines the salient
- **Candidate Answer Generation:** There are 2 type of search mechanism used, one is an using IR based search-engines like Lucene/Indry (unstructured search) and the other using human curated or auto-extracted KB (structured KB lookup). Unstructured search results in documents/passages which are not candidate answers. Titles or anchor text found in matching passages and their documents are used as answer candidates. Structure search yields a candidate answer directly
 - **Unstructured Search:** There are 3 kinds of searches performed
 - * **Document Search:** here we want to find documents which as a whole match the question. Here we use all the content words in the question to formulate a query (aka full-query)
 - * **Title-in-Clue Search:** Here the assumption is that if words in the question appear in title of a document, then the document is very likely to have the answer. So all documents with titles containing words from the question are aggregated and their passages are extracted for candidate answer generation. For this we use the full-query as well as LAT-based query (query containing LAT plus its modifiers)
 - * **Passage Search:** This is the traditional search which returns ranked passages matching keywords from the question. For this we use the full-query as well as LAT-based query
 - **Structured KB Lookup**
 - * **Prismatic:** Here we use LAT and is-a relationship to query PRISMATIC to retrieve all entities which are an instance of LAT.
 - * **Answer Lookup:** Here we use entity and relation(s) extracted from the parse of a question and use that to query a KB like DBpedia
- **Evidence Gathering & Scoring**
 - **Type Coercion**

- * **type-and-generate vs generate-and-type**

- * **Typing Sources**

- * **Type Checking Framework**

 - EDM

 - TR

 - PDM

 - TA

- * **Tycors**

 - Structured

 - Hand Crafted

 - Automatic

 - Unstructured

 - Hand Crafted

 - Automatic

- **Textual Evidence Gathering & Passage Scoring**

- * **Supporting Evidence Retrieval (SER)**

- * **Syntactic Semantic Graph**

- * **Passage Scoring Algorithm**

 - **Passage Term Matching Scoring**

 - **Skip-Bigram Scorer**

 - **Textual Alignment Candidate Scorer**

 - **Logical form Answer Candidate Scorer**

- * **Merging Scores across Passages**

- **Merging Scores & Ranking Candidate Answers**

2. Introduction to "This is Watson"

- **Paper:**

 - Introduction to "This is Watson" - IBM - 2012 - c161

 - * Anotated Copy

3. Question analysis: How Watson reads a clue

- **Paper**

– Question analysis: How Watson reads a clue - IBM - 2012 - c95

* Anotated Copy

- **Focus:** Is that part of the question s.t. when it is substituted by the answer, the question makes sense as a sentence,

– Example:

this man wrote treasure island

* here the focus is "this man"

- **LAT:** Is they lexical answer type which the question is seeking, e.g. *man* in the above question
- Focus and LAT are identified mainly using manual patterns (on text or dependency parse)

4. Deep Parsing

- Deep parsing is same as dependency parsing. It uses Slot Grammar to produce the parse as described below.
- **Slot Grammar:** Based on the idea of slots. Slots have 2 level of meanings:

– **Syntactic Roles:** Slots can be thought of as names for **syntactic roles** of phrases in a sentence. Examples of slot names are:

* subj

* obj

* iobj: indirect object

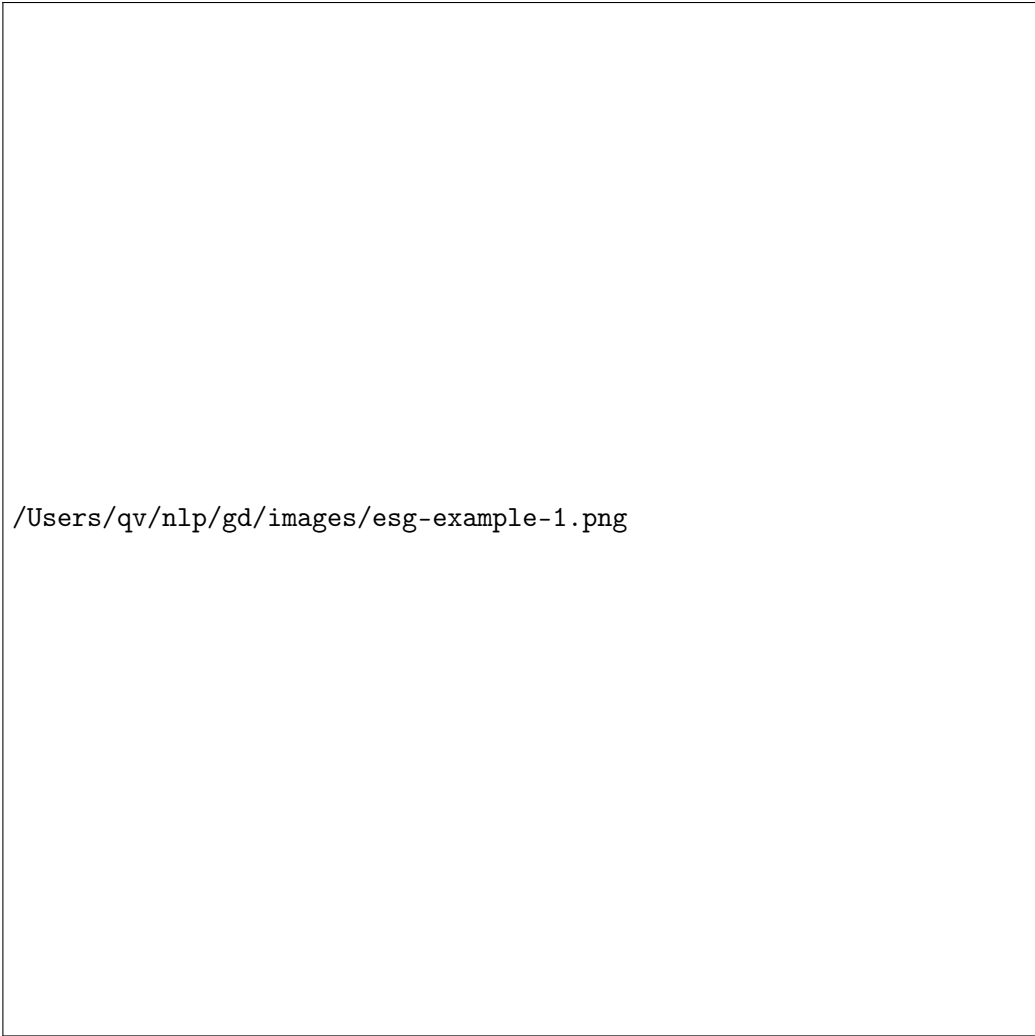
* comp: predicate complement

* objprep: obj of preposition

* ndet: NP determiner

– **Semantic Slots:** Some slots have a semantic significance. They can be thought of as names for argument positions for predicates that represent word senses.

* Consider the sentence:



/Users/qv/nlp/gd/images/esg-example-1.png

Here there is a word sense of *give* which, in logical representation, is a predicate, i.e.

$$give_1(e, x, y, z)$$

Where "e is an event where x gives y to z"

In this logical view, subj, obj, iobj can be thought of as names for arguments x, y and z respectively, i.e. these slots represent argument positions for the word sense predicate.

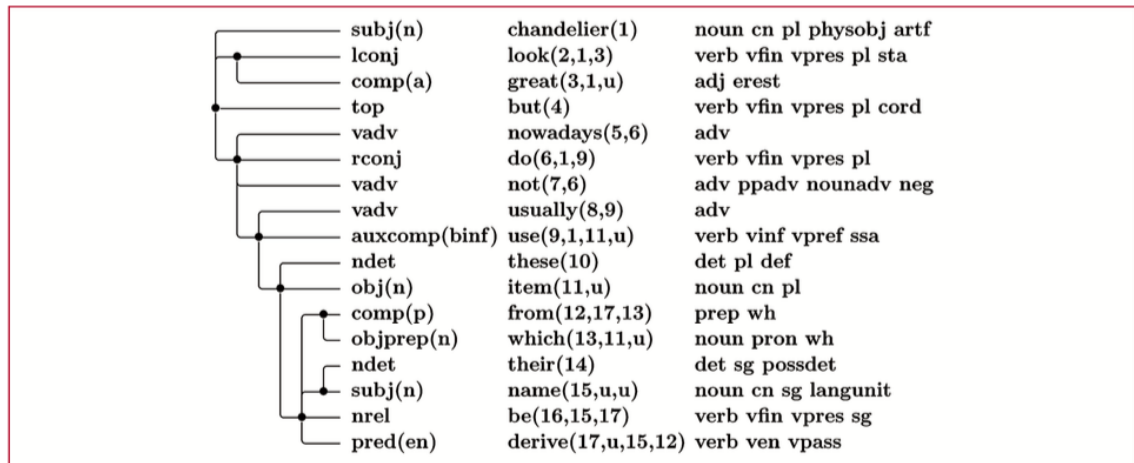
- * Slots which represent predicate arguments are called semantic slots
- * All other slots (i.e. ones that do not have semantic significance) are called **adjunct** slots. Example of adjunct slot in the above example is *ndet*. Adjunct slots are associated with parts of speech (POS) in the syntactic component of the slot grammar. (Seems like adjunct slot apply to POS of words rather than the word itself, i.e. they can go with any word with the same POS).
- * **Handling Syntactic Variations** Consider the following example which expresses the same logical proposition as the above example but is syntactically different:

Mary gave a book to John

The logical representation for this sentence is same as the previous one and uses the same complement **slot frame**, i.e. $give(\text{subj}, \text{obj}, \text{iobj})$.

- This **slot frame** comes from the same word sense entry for *give* in the lexicon.

- The syntactic analysis component of the Slot Grammar knows about these alternative syntactic ways of using slots - alternatives that lead to same logical representation
- Following paper gives a good introduction of slot grammar (specially section 2)
 - * Using Slot Grammar - IBM DeepQA - 2010
 - Annotated Copy
- **Features of SG analyses:** An SG parse tree is a dependency tree which expresses both semantic as well as surface (syntactic) structure. Each tree node N represents a (sub)-phrase of a sentence and is centered on a headword. This node's children in the tree are its modifiers, where each modifier M fills a slot in N. The slot shows the grammatical role of M in N. The lines and the first column on the left in the figure below show the surface structure while the centered column shows the deep (semantic) structure. The last column represents features which are syntactic as well as semantic. For more details read Using Slot Grammar .



The following are the features of a node in an SG parse:

- **Headword of the node:** The internal parse data structure stores several versions of the headword, including a) the form of it as it occurs in the text (inflected, mixed case, etc.); b) the lemma (citation) form; and c) the SG word sense of the node.

The headword is seen in the middle column as a predicate followed by arguments, for example, in *derive*(17, u, 15, 12).

We call this predication the word-sense predication for the node, and it is the main vehicle for showing the deep structure of the parse.

- **The ID of the node:** This is an integer that, in most cases, is the word number of the headword in the segment. This is 17 for the the headword *derive* in our example.
- **The logical argument frame of the node** In the internal parse data structure, this consists of the list of complement slots of the word sense, each slot being associated with its filler node (or nil if it has no filler). In the *derive* node of our example, this list of pairs would be (subj: nil, obj: ph15, comp:ph12), where ph15 is the phrase with node ID 15, spanning "their name", and ph12 is the phrase with ID 12, spanning "from which". The subj slot has no overt filler. Note that "derive" is given in the passive, but these three slot-fillers constitute the logical (active form) arguments of the verb. For example, ph15 is the logical obj of "derive", although grammatically, it is a subj (of "be"). That is why we speak of the logical or deep argument frame.

- * **Coordinated Conjunction Handling:** Note that, in the example, *chandeliers* (node 1) is shown as the logical subj of the predicates for *look*, *do*, and *use*, although in surface structure, its only role is as the (grammatical) subj of the coordinated node 4. In handling coordination, the SG parsing algorithm can "factor out" slots of the conjuncts. This happens with nodes 2 and 6, providing the common subj filled by node 1, but still showing 1 as logical subj for each conjunct.
- * **Implicit Arguments:** The predication which(13, 11, u), where *items* (11) fills the first slot (nsubj) of the relative pronoun *which*, is interpreted as showing that the relative pronoun is co-referent with *items*. Then, in building a logical form, the relative pronoun's variable can simply be replaced throughout with the variable for *items*.
- **The Features of the node:** Most of the features come from those of the headword sense, as obtained from morpholexical analysis of the headword
- **The Surface Structure of the node:** For each complement slot S, the slot option used for S is shown in parentheses after S. For instance, node 17, for *derived*, fills slot pred(en), meaning that node 17 fills a past-participial form of the pred (predicate) slot for *be* (in node 16).
- **Slot Grammar Lexicon:** A word corresponds to multiple word senses in the Slot Grammar Lexicon. Each word is associated with a sequence of sense frames, which lays down the slots it will have and constraints how those slots will be filled (e.g. an objprep slot can only take 'to' and 'for' preposition)
 - In other words, a slot frame is a template as to how a phrase (whose head word corresponds to the word sense of the frame) should be filled.
 - A word sense specifies the type and form of arguments (dependents) a word can take, its POS and other features for parsing.
 - The base ESG lexicon has 87K entries, but many more word forms are recognized because of derivational and inflectional morphology. It is also augmented by WordNet
 - Example of an entry in ESG base lexicon:
 talk < v (obj n (p about)) (comp (p to with)) < n obj1 (comp1 (p into)) < n nsubj (nobj n (p about)) (ncomp (p to with))
- **Predicate Argument Structure (PAS)**
 - ESG is used to create a dependency tree while PAS is modification of the ESG parse tree to normalize minor variations among dependency parses (thus facilitating writing fewer rules for relation extraction, Q&A alignment etc.
 - * For example, the passive form of an assertion results in the same PAS as the active one. In the ESG parse, the main passive verb carries the logical arguments which are same as for the active-form except that the logical subject might be a "by"-PP. PAS gets rid of the PP and replaces it with the object of the PP (in the logical subject slot).
 - Now PAS can use the same pattern to detect a relation over active as well as its passive variant. While ESG would require 2 variants of the same rule.
 - The following example sentences have small semantic differences which are reflected in ESG. The PAS outputs identical structure for all these.
 - * I heard that Edison invented the phonograph in 1877.

- * I heard that Edison invented a phonograph in 1877.
 - * I heard Edison invented the phonograph in 1877.
 - * I heard that Edison was inventing the phonograph in 1877.
 - * I heard that the phonograph was invented by Edison in 1877.
- ESG has two different dimensions of structure: deep structure and surface structure. PAS collapses these two dimensions into one. The structure of PAS generally follows the deep structure, except for those types of slots that are encoded only in the surface structure
 - A variety of nodes from ESG are omitted from the PAS. Links to and from these nodes in the parse tree are instead redirected to related nodes in the PAS. For example, links to/from auxiliary verbs are instead treated as links to/from the corresponding main verbs. The following types of nodes are dropped by PAS:
 - * Auxiliary verbs
 - * infinitive "to" marker and the "that" which introduces "that" clauses.
 - * Determiners except for those one which are high semantics, e.g. possessive pronouns and negation
 - * Forms of "be" with no predicate (or for which the predicate is adjective)

• **Uses:**

- Relation Extraction: identifies semantic relationships among entities in a sentence using the results of deep parsing, e.g. *authorOf*, *actorIn*, *bornIn* etc.
 - * PAS/ESG systematically identifies and explicitly labels long-distance relationships between constituents, thus bringing even more diverse text forms closer together. Thus it facilitates structural patterns as opposed to text patterns.
 - * For example, the following structural pattern would match a large number of sentences expressing *authorOf* relation (with very diverse text)

authorOf :: [WriteVerb] -> subj -> [Author] & [WriteVerb] -> obj -> [Work]

Here *WriteVerb* could be words/phrases expressing the event of writing, e.g. 'wrote', 'penned', 'author of' etc. [Author] is a noun/pronoun entity (detected by NER/WordNet?) and [Work] is the title, e.g. 'War and Peace' (again detected by NER/WordNet?). The 'subj' and 'obj' are dependency link label.
- Question Analysis: to identify the LAT the question is seeking. Consider the following example:

"Chandeliers look great but nowadays do not usually use these items from which their name is derived"

In this example the focus is "these items" which is inferred based on the fact that ESG parsed "these" as a determiner. In this example the LAT is "items" which is inferred from the ESG parse which marks "items" as the head word of the focus phrase. LAT is usually the headword of the focus phrase or a word closely linked to the focus in the dependency tree.
- Keyword Search Component: identifies keywords which have a strong semantic relation with the focus phrase in the question and give those higher weight in the search query

- Passage Scoring Component: uses the result of deep parsing on both the question text (with focus replaced by candidate answer) as well as a passage with candidate answer to determine if their respective dependency trees align around the candidate answer.
 - * In our example, the PAS shows that the focus, "these items", is the object of the verb "use", which has subject "chandeliers" and modifier "not". If a passage contains the verb "use" with the object being a candidate answer, the subject being "chandeliers", and a modifier "not", then these passage scorers will assign a very high score to this candidate answer.
- Type Coercion: uses PAS to compare the LAT in the question with the candidate answer in the passage
 - * One of the many sources used for typing information is Wikipedia categories, which are short text strings describing the type of entity that the page is about.
 - For example, the Wikipedia page for "Toronto, Ohio" has as one of its categories "Populated places in Jefferson County, Ohio". Given this label, we would want to conclude that "Toronto, Ohio" could be a legitimate answer to a question asking for a "place," but we would not want to conclude that "Toronto, Ohio" could be a legitimate answer to a question asking for a "county" (although both of these words appear in the category text). The ESG parse shows "places" as the syntactic head of the NP "Populated places in Jefferson County, Ohio." Consequently, our type coercion algorithms can conclude that this category is asserting that Toronto, Ohio, is a place but would not conclude that Toronto, Ohio, is a county
- Prismatic KB creation: The PRISMATIC knowledge base includes statistics relating to how often sets of words are connected to each other via specific PAS structures and semantic relations. This KB in turn is used by a variety of search as well as scoring components.
 - * For example, PRISMATIC has a record of how often the word "chandelier" is the subject of the verb "use" in a large corpus and also of how often any given word is the object of the verb "use" with subject "chandelier". This data is used for finding and evaluating candidate answers.
- **Question:** Can we use the results of a dependency parse (from a shift reduce parser like syntax-net) to mine semantic relations efficiently (in # of rules) similar to DeepQA's use of PAS. The point of the question is whether we can build something like PAS on top of syntax-net which will normalize away lexical variations across the dependency parses of the same logical preposition and hence facilitate writing of a very small # of rules per cent semantic relation.
- **Paper:**
 - Deep parsing in Watson - IBM - 2012 - c65
 - * Annotated Copy

5. Textual resource acquisition and engineering

• Introduction:

- To support a high quality QA system, it needs high quality information resources. To acquire and adapt the resources for a give QA task, the following needs to be done.

- **Source Acquisition**

- Consists of initial domain analysis to identify what baseline resources would be useful for the domain addressed by a given QA task (Jeopardy in this case).
- Evaluation Measures:
 - * Accuracy: % of Qs answered correctly
 - * Binary Recall: whether a Q's answer is present in the candidate passages selected after an IR search on the information corpus
 - * Precision@70: precision if only top 70% Qs are answered if Qs are ranked based on confidence score of their answer
- **Initial Domain Analysis:**
 - * Answers to Jeopardy Qs were titles of Wikipedia docs in 95% cases making the later a good baseline source
 - * Post IR search, only 77.1% of the Qs had correct answers among the candidate passages generated. Which means there was no coverage for quarter of the Qs in the baseline corpus. So this gap needs to be covered as addressed below.
- **Covering Gaps:**
 - * Encyclopaedias, Bible, Works of Shakespeare, Famous books from project Gutttenburg, Dictionaries, Wiktionary etc.

- **Source Transformation**

- **Converting Non-Title Oriented to Title Oriented Documents**
 - * **Important Property of Title Oriented Docs**
 - If a Qs specifies a lot of properties of the answer and the text in a title-oriented document matches well with the Q, then the title is likely to be the answer.
 - If a title of a document matches entities or non-stop words in the Qs, then the answer is likely to be in that document.
 - * Based on the above 2 properties of the document, answer retrieval strategies can focus on the following 3 methods:
 - **passage search:** this is the traditional strategy based on extracting relevant short passages from a corpus adopted by existing QA systems
 - **Document Search** - Based on property 1 above
 - **Title-in-Clue passage search (TIC)** - based on property 2 above
 - * The results with the additional document and TIC searches were significantly better than just passage search. This indicates that we should convert non-title oriented documents to title-oriented documents

* **Generating Title Oriented psuedo-Docs**

- Some of non-title oriented docs are Shakespeare's work, bible, song lyrics etc.
- non-title oriented sources are converted to title-oriented based on content of each source and the likely relationship will be sought b/w the content and potential answers.
- For example, a character-centric set where the lines spoken by each Shakespeare play character is extracted to form a document whose title consists of the name of the character
- **Salient Topics** There is value in automatically identifying salient topics from a non-title oriented document and create psuedo-documents with the topic as the title and topic relevant sentences as the content.

– **Reorganizing Title Oriented Docs:**

- * **Merging documents** which are essentially same but with different spellings
- * **Updating definitional sentences** to contain the term that is being defined.

- For example, in a dictionary, each entry can be considered as a document with the headword as the title. Here, the document text contains the definition of the headword. The definition itself doesn't contain the headword (i.e title). This is fine for primary search but not good for candidate scorers. For example, logical scorer which tries to find the strength of alignment (i.e. similarity) b/w the dependency parse of the question and the candidate passage doesn't work well because the candidate passage doesn't include the candidate answer. It just contains the definition.

For example the definition of 'program' just contains:

- A set of structured activities.

The logical scorer will perform better if the definition is update to include the head-word, i.e.

- A *program* is a set of structured activities.

• **Source Expansion:**

- Useful for increasing coverage and lexical+syntactic variation on the content of the source documents by selectively acquiring additional knowledge from the Web.

- * Lexical and syntactic variations facilitate the extraction and the validation of answers because if the same information is conveyed in different ways, this increases the chances that the system can successfully match one representation against the question

- **Selecting Seed Documents:** More popular documents are more likely to have answers to Qs (in Jeopardy, this was borne out by plotting the # of documents containing answers vs. the # of top documents by popularity and this graph grows fast intially and then plateaus out at around 300K wikipedia docs). So the top popular documents are used as the seed set.

– **Retreival:**

- * Use the title from the seed set of docs to perform Google Search

- select top ranked results (say top 100)
- **Extraction of Text Nuggets**
 - * Splits each retrieved document into paragraph-length nuggets
 - * A nugget should ideally be self-contained and be either entirely relevant (to the title from a seed doc which was used in initiating the search) or non-relevant.
 - Longer nuggets are self contained but only partially relevant
 - Nuggets based on structural markup represent a good tradeoff b/w the 2 properties
- **Scoring & Ranking the Nuggets**
 - * Done by a statistical model which scores relevance of extracted nuggets wrt topic of the seed document
 - * Based on binary classification (relevant vs. none-relevant using logistic regression)
 - * Features:
 - Topic similarity: using tf-idf weighted cosine similarity b/w seed document and the nugget) or other language modelling techniques (e.g. comparing word vectors)
 - Search features: e.g. search rank
 - Surface features: lexical features like nugget length
- **Merging:** first the ranked nuggets are filtered for redundancy and then merged into a psuedo-document
 - * nuggets whose keywords are entirely subsumed by seed or higher ranking nuggets are dropped
 - * Nuggets with relevance score below a threshold are also dropped
 - * If total character length of all nuggets exceed a threshold relative to the length of seed document

- **Paper**

- Textual resource acquisition and engineering - IBM - 2012
 - * Annotated Copy

6. Automatic knowledge extraction from documents (PRISMATIC)

- **Introduction:** The main idea is to use dependency parse to create frames. A frame represents shallow knowledge. Frame projection is used to induce knowledge. Frame projections are nothing but rules defined on frames. Aggregate Statistics is used to identify frames (or parts of frames) which occur regularly (i.e. patterns or frame projections) in the corpus. These can be used to learn binary relations between entities, entity types etc.

- We can learn assertional axioms, i.e. facts, e.g. "the best known thing that Einstein won is a Nobel Prize". This is known as extensional knowledge (i.e. the entities are instances (proper nouns) and the knowledge is not general)
- We can learn terminological axioms e.g. "scientists publish papers" or "scientists win Nobel Prizes". This is known as intentional knowledge (where entities are types or common nouns and the knowledge is general)
- **Framing:** involves extraction of frames from a dependency parse tree.
 - **Frame:** basic semantic unit representing a set of entities and their relations found in a piece of text. A frame is composed of a set of slot-value pairs.
 - * Frame is intentional if its values are types
 - * Frame is extensional if its values are instances
 - **Slot:** A slot is a binary relation which are dependency relations in a parse tree (same as arc labels in the tree), e.g. predicate, subject, object, prepositional modifier of a noun etc.
 - **Slot Value:** lemma form of a word from the sentence or a type annotated by NER
 - **Frame Projection:** A portion (subset of slot-value pairs) of a frame which occurs with regularity and forms a pattern. For example the subset containing just the subject, verb and object (i.e. S-V-O projection) is useful in analyzing the selectional preference of verbs.
- **Framing Process**
 - **Corpus Processing**
 - * Dependency Parsing: uses ESG
 - * Co-reference resolution: uses rule-based co-reference resolution
 - * Relation Detection: identifies is-a/subtype-of relations
 - **Semantic Annotation (NER)**
 - * Rule based NER is used to identify types of slot-values. Useful for producing terminological axioms (using ontological generalization)
 - **Frame Extraction**
 - * Frame elements are taken from direct arguments of a predicate.
 - * Each frame is restricted to two levels of the parse tree.
 - * Semantic relations like is-a slot is added using a syntactic pattern-based relation detector
 - * NER annotations (i.e. type for a slot value) are also included
 - * Proper Noun flag is also included for each slot entry
 - **Frame Projection:**
 - * Projection operation on a frame is defined as taking & analyzing a portion (subset) of a frame

- For Example, an (N-P-OT) projection only keeps the Noun, Preposition & Object-type slots of a frame and discards the rest.
- Examples of Frame Projections:
 - S-V-O, S-V-O-IO, S-V-P-O, N-Isa, N-Isa-Mod etc. (S = Subject, V = Verb, O = Object, IO = Indirect Object, N = Noun, Mod = Modifier)
- Examples:
 - V-OT projection can be use to induce knowledge such as what are "annexed" are "regions". This can be used in "Napolean annexed Piedmont in 1921" to infer that 'Piedmont' is a region.
 - Using N-Isa projection, we can see multiple frames of the form $\{\langle \text{noun: Bill Clinton} \rangle \langle \text{isa, politician} \rangle\}$. This allows us to learn that Bill Clinton is a politician.
 - Facts (assertional axioms) are learnt using S-V-O projections where both S and O are constrained to be proper nouns, e.g. in "USA annexed Texas from Mexico", we learn the binary relation $\{\langle \text{subject: USA} \rangle \langle \text{verb: annex} \rangle \langle \text{object: Texas} \rangle\}$.

• Aggregate Statistics

- **Frequency:** $\#$ of times a particular frame occurs $\#(f) = |\{f_i \in P | \forall_{s \in S} V(s, f) = V(s, f_i)\}|$
 - * where P is the bag of projections found in a corpus
 - * Frequency gives us the an estimate of popularity of fillers for a given set of slots
 - * For example, the frequency of $\{\langle \text{subject: Einstein} \rangle \langle \text{verb, win} \rangle \langle \text{obj, award} \rangle\} = 142$ may suggest that Einstein winning awards is a set of popular fillers in the corpus.
 - * Frequencies are affected by size of corpus and popularity of individual filler
- **Conditional Probability**
 - * $p(f|f') = \#(f)/\#(f')$, where $f' \subseteq f$
 - * For example, let $f = \{\langle \text{subject: Einstein} \rangle \langle \text{verb: win} \rangle \langle \text{object: award} \rangle\}$, and $f' = \{\langle \text{subject: Einstein} \rangle \langle \text{verb: win} \rangle\}$ then $p(f|f') = 0.32$ which means that 32% of the time when Einstein wins something, he wins an award
 - * Not affected by the size of corpus and so it is easy to compare one conditional probability with another
- **Normalized Point-wise Mutual Information**

$$npmi(f, f') = \frac{pmi(f, f')}{-\ln \frac{\max(\#(f), \#(f'))}{N}}$$

where $pmi(f, f') = \ln N \frac{\#(f \cup f')}{\#(f) \times \#(f')}$

where N is the size of a particular frame projection

 - * Compared with conditional probability, NPMI takes into consideration the popularities of different subsets of a frame and their correlation.
 - * For example, let $f = \{\langle \text{object: award} \rangle\}$ and $f' = \{\langle \text{subject: Einstein} \rangle \langle \text{verb: win} \rangle\}$; if $npmi(f, f') = 0.7$, then it indicates a high degree of co-occurrence between Einstein winning and award.

- **Application in Watson**

- **Type Coercion**

- * Used to determine whether a candidate answer is of LAT expressed by the question. We can use aggregate statistic of {<noun: candidate answer><isa: LAT>} frame to determine the likelihood of the candidate answer being of type LAT.
 - * For example, the Q "Senator Obama attended the 2006 groundbreaking for this man's memorial, 1/2 mile from Lincoln's" has LAT as man. Candidate search will yield MLK as a potential answer. We can check if that MLK matches LAT by checking if {<noun: MLK><isa: man>} frame projection has significant aggregate statistics.

- **Candidate Generation**

- * Various projections can be used to generate candidate answers
 - * For example, in Q
"While Maltese borrows many words from Italian, it developed from a dialect of this Semitic language".
Here the LAT is language with modifier Semitic. Aggregate statistics on the frame projection consisting of isa slot with value "language" and modifier slot with value 'Semitic' is used to determine the top 20 most common instances of Semitic language, which include the correct answer Arabic.
 - * Another Example Q:
"To: junglechick. From: mr_{simian}. Subject: pronouns. Fyi, I do use 'I' when referring to myself in a 1914 novel" with category "LITERARY CHARACTERS' E-MAILS".
Here the LAT is character with modifier literary. 'Tarzan' the correct answer is the 6th most popular literary character based on aggregate statistics in the frame-projection based KB.
 - * Answers produced by frame projection are more likely to be correct than other candidate generation schemes

- **Missing Link**

- * Frame projections can be used to identify semantic relations between entities. Some jeopardy Qs require finding a hidden entity which is connected to entities in the Qs.
 - * For example, for Q "On hearing of the discovery of George Mallory's body, he told reporters he still thinks he was first", the entity Mount Everest is not stated, it provides the common link between George Mallory and the correct answer Edmund Hillary. We will find frames in our KB consisting of {<subject: George Mallory> <verb: perish> <verb preposition: on> <object preposition: Mount Everest>} and {<subject: Edmund Hillary> <verb: climb> <object: Mount Everest>} which identifies the connection between Hillary and Mallory.

- **Type Inference and its uses**

- * Frame projections (with aggregate statistics) can be used to learn the predominant types of subject & object arguments for verb and noun phrases. This knowledge can be used to infer types of named entities in a new piece of text. The inferred types can be used for either determining the LAT (in Qs where the LAT is not obvious) or in checking whether the type of a candidate answer matches the LAT.

- Example: From "Napoleon annexed Piedmont in 1859" and the frequently occurring frame {<verb: annexed><isa: region>} we can conclude that Piedmont is most likely to be a Region.
- * We can infer fine grained knowledge for a context.
 - For example, the sentence "He ordered a Napoleon at the restaurant". A dictionary-based NER is very likely to label Napoleon as a Person. However, we can learn from a large amount of data that in the frame {<subject-type: Person> <verb: order> <object-type: ?> <verb-preposition: at> <object-preposition: restaurant>}, the object type typically denotes a Dish and, thus, correctly infer the type for Napoleon in this context.
- * Type inference can also be used in heuristics to detect LAT of a Question.
 - We can use the most frequent lexical type values for the object slot and use that as potential LATs. However, a key issue is taking into account the larger context (or topic) of the relation, without which the system can generate incorrect or noisy results.
 - For example, in Q "In the billiards game named for this black object, you must sink it last", the focus is "it" and its LAT can be determined by finding the most frequent lexical type values for the object slot of "sink". But typically "sink" goes with "ship" and not "ball" which is the correct answer.
 - The problem lies in the lack of context-aware knowledge in frame-projection
 - Instead, a better idea is to check whether a candidate answer goes well as the object slot of the verb sink.
 - The idea is that because candidate answers are typically generated on the basis of the context of the question (Billiards in this case), checking whether a given candidate fits into the frame of the question using projection (i.e. has significant aggregate statistics for the projection containing 'ball' as object of verb 'sink') is an easier task than predicting the correct LAT for the question, which requires considering context

• Future Work

- **Context:** The idea is to take context into consideration when producing the aggregating statistics
- **Coverage:** Increase coverage using better co-reference resolution component and larger corpus
- **Confidence:** Because none of the NLP components is perfect, it is better to include confidence values in frames that incorporate the scores of each of the NLP components' outputs. The confidence score may help create better aggregate statistics than the ones based only on the frequency count.

• Paper:

- Automatic knowledge extraction from documents - IBM - 2012 - c82

* Annotated Copy

7. Finding needles in the haystack: Search and candidate generation

- **Introduction:** To produce a list of possible answers, Watson uses a 2 phased approach. First it does search to retrieve relevant documents and passages. Second, it generates candidate answers from the retrieved content. It differs in 2 ways from traditional QA systems which only do passage retrieval. One, it extends passage search with title-in-document as well as title-in-clue search strategies. Two, it uses structured KB in addition to unstructured data sources.

In the discussion below we use the following Jeopardy Question as an example:

MOVIE-"ING" Robert Redford and Paul Newman starred in this depression-era grifter flick. (Answer: The Sting)

- **Unstructured Search**

- **Query Generation for Unstructured Search** Two types of queries are generated.

- (a) Full Query: based on the content words and phrases in the Question
(2.0 Robert Redford) (2.0 Paul Newman) star depression era grifter (1.5 flick)
- (b) LAT-only Query: For the case where LAT has modifiers. The query contains LAT and its modifiers
depression era grifter flick

- * This type of query is used because in some cases the LAT and its modifiers uniquely identify the answer and in other cases narrow the answer candidate space significantly.

- **Search Strategies**

- * Three pronged approach as described below. DeepQA extends traditional strategy of passage retrieval with 2 new strategies (Document Search & TIC search). It also uses multiple search engines to increase recall.

- * **Document Search:** Based on the observation that a large # of answers to Jeopardy Questions are titles of Wikipedia Documents. It introduces a document search strategy where the content of the document match well with the properties of the answer described by the Question and the answer is the document title itself. Here the idea is that the content of the overall document match well with the Question instead of a single or few passages.

- Only full query (described above) is used for document search.

- * **Title in Clue Search:** Based on the observation that if the title of the document is present in the Question then the answer is contained in a passage in the document. This search strategy focuses on a small number of documents whose title match word(s) in the Question. This strategy yields a low # of candidates and has a high precision.

- To identify the documents whose title is contained in the Question a dictionary which maps canonical Wikipedia titles (e.g. Naomi) to all Wikipedia documents with that string as their title (e.g. Naomi(band) and Naomi (title)). The dictionary is used to identify document titles in the question, and a subcorpus of documents is dynamically constructed by aggregating all target documents in the matched dictionary entries.

- * **Passage Retrieval:** This is the traditional strategy where the title of a document is neither the answer nor is it present in the question. In that case the traditional passage retrieval is used (as used by other QA systems).

- **Search Engines:** Two search engines, Indry & Lucene are used with certain modifications.
 - * The two are complementary as they use different ranking strategies. **Indry** uses a combination of language modeling and network inference. While **Lucene** uses TF-IDF for ranking.
 - * **Indry:** Used for document Search as well as passage search.
 - * **Lucene:** Only used for passage search. Lucene was modified to use a two-step passage retrieval. First, relevant documents matching the query are retrieved. Second, passages are extracted from these documents and ranked based on the following features:
 - Query-independent features:
 - Sentence Offset: sentences closer to the beginning of document are more likely to be relevant.
 - Sentence length: larger sentences are more likely to have the answer.
 - Number of Named Entities in the passage: anchor text and document titles are used as proxies for named entities.
 - Query-dependent feature: a measure of keyword and phrase matching

- **Structured Search:**

- **Answer Lookup:** Query is translated into a formal lookup in DBpedia or IMDB. A relation r and a named-entity $e1$ are identified from the Question and used to find a tuple of the form $(e1, r, e2)$, where r related $e1$ in the question with $e2$ which is the answer.
 - * Manual rules (grammar rules) are used to extract these relations. Since these rules are hard to define, we focus only on the relations that most occur in Jeopardy Questions
 - * This method's contribution to overall candidate answer generation is minuscule.
- **Prismatic Search:** One of the semantic relations Prismatic records is the occurrence of is-a relation. This relation is particularly useful for search and candidate generation because it can identify the most popular instances of a particular LAT.
 - * For example, in Q

"While Maltese borrows many words from Italian, it developed from a dialect of this Semitic language".

Here the LAT is "language" with modifier "Semitic". Aggregate statistics on the frame projection consisting of isa slot with value "language" and modifier slot with value "Semitic" is used to determine the top 20 most common instances of Semitic language, which include the correct answer "Arabic".

- **Candidate Answer Generation:**

- Document Title: is used as a candidate for documents retrieved in document search
- Wikipedia Document Title: in case of passage retrieval, words in the passage which match Wikipedia Document Titles are used as candidate answers

- Anchor Text: passages retrieved from Wikipedia articles contain meta-data like anchor text, document titles in hyperlink targets etc. which can be used as candidate answers.

- **Results of Combination of All Search Strategies**

- Binary Recall: 87.17%
- Accuracy: 71.32%
- three-fourth of the 429 questions with candidate recall failures, roughly three-fourths of them are due to search failures.
- For the other one-fourth, search returned a relevant passage, but Watson’s candidate generation strategies failed to extract the answer as a plausible candidate. Most of them were verbs or common nouns which have low coverage in titles as well as anchor texts.

- **Paper:**

- Finding needles in the haystack: Search and candidate generation - IBM - 2012 - c61
 - Annotated Copy

8. Typing candidate answers using type coercion

- **Introduction:** Tycor is an approach used for finding evidence whether a candidate answer type is consistent with the LAT in the corresponding Question. DeepQA includes a logical framework for **TyCor** with a variety of instantiations (aka components) using an assortment of structured (e.g. DB-Pedia/YAGO) and unstructured (e.g. Wiki-Intro) sources for typing. Tycor is used in DeepQA as a form of evidence scoring; it is performed after answers are generated and before they are ranked.
 - Tycor is implemented as multiple components. The components use a common logical framework (not concretized in software yet) but with different sources of typing information and different algorithms/strategies.
- **Answer Types:** Jeopardy Qs have a very wide variety of LATs with a very long tail of types. Roughly 5K different type words were used in 20K Qs. More than 1/2 of the types were used less than 3 times and 12% are used only 1 times.
 - This leads to the conclusion that the typical strategy used by traditional QA system of type-and-generate, i.e. first determine the answer type from the Q and then only allow candidates of that type is brittle for 2 reasons
 - * A fixed number of answer types is brittle as new types are being introduced constantly
 - * Sometimes it might be hard to find evidence that the correct candidate answer is consistent with the answer type expected by the Q.
 - Rather than type-and-generate, DeepQA takes a strategy of generate-and-type. The later means that we first generate the candidates and then judge whether they are consistent with the LAT.
 - * The idea is to try several different strategies to coerce candidate answer type to be consistent with the LAT and not reject the candidate even if strong evidence of type consistency with LAT is not found.

- * Instead, we just compute a score about how confident we are that the candidate answer type is consistent with LAT and pass it to the final answer merging and ranking phase.

- **Tycor in the DeepQA architecture:**

- Tycor uses LAT as input from Question Classification stage
- Tycor uses candidate answers generated by the candidate generation stage as the other input.
- **Tycor** components are a subset of hypothesis and evidence gathering
- **Final-answer merging and ranking:** The TyCor scores are among the many features used by this component.

- **Tycor Logical Framework** Consists of 4 parts

- **Entity Disambiguation and Matching (EDM):** Maps candidate answer to an entity, e.g. Napoleon is an Entity
 - * EDM must account for both polysemy (the same name may refer to many entities) and synonymy (the same entity may have multiple names)
- **Type Retrieval (TR):** Maps entity to a type
 - * e.g. NED maps 'Napoleon' to type 'PoliticalLeader' while YAGO maps it to 'Emperor'
 - * In unstructured sources, this may require parsing or other semantic processing of the natural language.
- **Predicate Disambiguation and Matching (PDM):** Maps LAT to a source specific type
 - * e.g. NED maps 'Emperor' LAT to 'PoliticalLeader' type
 - * PDM strongly corresponds to notion of word-sense disambiguation with respect to a specific source
- **Type Alignment:** aligns type extracted from LAT with type extracted from candidate answer.
 - * Note that EDM, TR and PDM could have generated multiple entities and their types. TA tries to find the best alignment between the types extracted for the candidate-answer and the types extracted for the LAT. This is nothing but a confidence score which is passed to the final-answer merging and ranking stage
 - * Uses various strategies to look for subsumption and disjointedness, e.g. hierarchical ontologies, disjointedness lists, is-a relation in KBs, dependency parsing and alignment in case of types being natural language etc.

- **Typing Sources:**

- **Categorization:** Sources can be categorized in different ways
 - * **structured** or **unstructured**
 - * Creation based: **community generated**, **handcrafted** or **automatically extracted**

- **Useful Properties:** Some useful properties which are used in type checking are:
 - * Hierarchical ontology with **hyponym** and **hypernym** (instance-of) links used for checking subsumption, e.g. 'Napolean' is an instance of 'Emperor'.
 - * **Disjointness constraints**, e.g. a 'Person' is not a 'Country' etc. Used for generating negative evidence feature (i.e. that types are not consistent)
 - * Typing relations like **is-a** relation automatically extracted from data source like Wikipedia and stored in KBs.
 - * **Gender** determined from pronouns used in data corpus
- Here we provide a list of sources by categories and some of their properties
- **Structured:** these define formal entities and types, sometimes as a hierarchical ontology (e.g. YAGO) and sometime flat (e.g. NED).
 - * **Broad Community Generated:**
 - **DBpedia/YAGO:**
 - many candidate answers in DeepQA are generated from titles of Wikipedia documents. These titles are entities in DBpedia. DBpedia is a linked open data-source automatically compiled from Wikipedia infoboxes and article templates.
 - Entities in DBpedia have types represented in RDF from YAGO. YAGO is a semi-automatically created type taxonomy based on WordNet, Wikipedia & corpus analen-title
 - DBPedia is used for mapping candidate answer to entity
 - YAGO is used for TA (using subsumption)
 - **Wordnet:** hierarchical ontology
 - Used to check for consistency of candidate answer type with LAT using the hypernym/hyponym relations.
 - Also Contains Biological & Geographic Taxonomies used for mapping candidate answers to entities
 - * **Hand-crafted:**
 - **ClosedLAT** certain LATs identify types with enumerable lists of instances, such as countries, presidents, US states etc.
 - **NED:** Rule based Named Entity Detector with 100 structured types
- **Unstructured**
 - * **Broad Community Generated:**
 - Wiki-Category: Wikipedia articles are frequently tagged with explicit categories in the form of natural-language text. All of these categories are stored in DBpedia.

- Wiki-List: Wikipedia and other web sources compile list of things associated in some ways, e.g. list of 'Argentinian Nobel Prize Winners'.

* **Automatically Generated**

- Wiki-Intro: uses introduction sentence in Wikipedia Articles
 - Prismatic: relations extracted from Wikipedia and stored in a KB using pattern based relation extraction technique.
- **Tycors:** Tycor is implemented as several components each implementing its own type coercion strategy. The following table describes the various components.
 - **Handling types which are strings:** Note that certain tycors produce natural text string as types (unlike others which use formal types from ontologies). For TA (type alignment) in this case, we first do dependency parsing of the types to identify head words of phrases etc. and then use ontological resources like WordNet/YAGO to compare those. For example Wiki-List produces 'French Monarch' as the type for 'Napolean' while the LAT is 'Emperor'. It extracts 'Monarch' as the head word in 'French Monarch' and matches it to 'Emperor' using WordNet Ontology.

Tycor	EDM	TR	PDM	TA
YAGO	DBpedia lookup	YAGO	YAGO	- Check Subsumption using YAGO ontology - Negative Evidence using Disjointedness Lists
WordNet	WordNet	WordNet Ontology	Wordnet Ontology	- Subsumption using WordNet Ontology
ClosedLAT	candidate answer	N/A	N/A	check if candidate answer in ClosedLAT list derived from LAT LAT
NED	NED on candidate answer	NED types	NED on Q	Compare the Q NED type to that of answer
Gender	candidate answer	using pronouns referring to candidate answer in corpus	from LAT	Compare the extracted Genders
Lexical	candidate answer	N/A	N/A	Algorithms to check constraints indentified in LAT, e.g. answer should be verb, phrase or first name
Wiki-Category	candidate answer	candidate answers from Wikipedia tagged with category	LAT itself	Handling Types which are Strings
Wiki-List	candidate answer	Find matching LAT list. Type is part of list name	LAT itself	Handling Types which are Strings
Wiki-Intro	candidate answer	Lookup relation type KB (which is extracted from introductory sentence from every Wikipedia article)	LAT itself	Handling Types which are Strings
Passage	candidate answer	occasionally the passage itself asserts candidate's type	LAT itself	Handling Types which are Strings
Prismatic	candidate answer	relation type KB extracted from Wikipedia articles by detecting is-a relation	LAT itself	Handling Types which are Strings
Identity	candidate answer	asserted by answer, e.g 'the chu river' is a river	LAT itself	Handling Types which are Strings

- **Example:** Let us see how each TyCor handles the following example:

Consider a question asking for an 'emperor' and the candidate answer 'Napoleon'.

– **EDM:**

- * YAGO: Looks up 'Napoleon' in DBpedia and finds several matches, e.g. dbpedia:Napoleon which is a strong match. It also finds dbpedia:Napoleon_%28cardgame%29 which is a weak match (assigned a lower score). dbpedia:Napoleon is linked to type 'Emperor' in YAGO.
 - In reality, we map the candidate answer string to a Wikipedia page using a variety of resources such as Wikipedia redirects, extracted synonym lists, and link-anchor data. We then use the title or anchor text there to lookup DBpedia/WordNet.
- * WordNet: Looks up 'Napoleon' and finds 3 different senses
- * All others: simply uses candidate answer as entity

– **TR:**

- * YAGO/WordNet: Those TyCor strategies that identify a formal entity [e.g., a DBpedia URL or a WordNet synset (or set of synonyms)] during EDM are generally able to look up one or more types for that entity in some structured source. Here 'Napoleon' is mapped to 'emperor', 'leader' etc.
- * NED: maps 'Napoleon' to 'PoliticalLeader'
- * Wiki-Category: Uses categories tagged on wikipedia articles with title 'Napoleon' as types, e.g. 'Monarch', 'Emperor' etc..
- * Wiki-List: Maps it to 'French Monarch' as 'Napoleon' is part of the list of 'French Monarchs'
- * Wiki-Intro/Prismatic: Looks up 'Napoleon' in a relation typing KB to identify it as 'Emperor', 'King' etc.
 - The relation KB was populated from sentences like "Napoleon was a french Emperor" or "The lion (Panthera leo) is one of the four big cats in the genus Panthera".
- * Passage: Maps 'Napoleon' to several types including 'Emperor' and 'PoliticalLeader'.
 - Occasionally, passage containing the candidate asserts the candidate's type, e.g., 'Christian Bale' is the first actor to really do Batman justice[. Passage TyCor uses pattern-based relation detection to identify assertions that some entity has some type and attempts to match the asserted type to the LAT
- * Identity: Uses phrases like 'Emperor Napoleon' in the corpus.

– **PDM:**

- * YAGO/WordNet: Those TyCor strategies that use formal types lookup these types in some ontology. In this example YAGO maps 'Emperor' to 'emperor' type.
- * NED: identifies 'emperor' in the Question as an entity of type 'PoliticalLeader'
- * All others: simply return LAT string as a type

– **TA:**

- * WordNet/YAGO/NED: For TyCors that produce formal types in a type hierarchy from TR and PDM, the type alignment process involves checking for subsumption, disjointness, etc.
- * All others: See **Handling Types which are Strings** above. For example, Wiki-List tries to determine whether it can conclude that a "French monarch" is an "emperor" by parsing both (e.g., finding that "monarch" is the headword of "French monarch") and matching terms using resources such as WordNet and Wikipedia redirects.

• **Papers**

- Typing candidate answers using type coercion - IBM - 2012 - c59
 - * Annotated Copy
- Question-answering by predictive annotation - IBM - 200K - c234

9. Textual evidence gathering and analysis

• **Introduction:**

- DeepQA, after generating a set of candidate answers (CA), employs a diverse set of evidence-scoring components to quantify evidence relating to each CA. One type of evidence are passages containing the CA.
- This paper presents the mechanisms for collecting and scoring these passages.
- The collection mechanism is known as Supporting Evidence Retrieval (SER). The idea behind SER is to insert a CA back into the original question to form a preposition; and then run DeepQA search (as presented in the 'Finding needles in the haystacks' paper) to find passages most closely related to this preposition.
- This paper also presents four passage-scoring algorithms described below.
- There are 2 core ideas behind this paper:
 - * Analyzing passages using a variety of scoring strategies, at different depth of analysis, is significantly more effective than a single strategy
 - * SER improves effectiveness of passage scoring by providing more passages for each answer than are found in primary search.

• **Supporting Evidence Retrieval (SER)**

- Search Query is build by combining CA (required) with one or more terms from the Question.
 - * Uses Indri as a passage search engine and gives more credit to passages that match more Question terms. The 20 highest ranked passages are sent to the scoring algorithms.
- **Example:**

"In 1840 this German romantic married Clara Wieck, an outstanding pianist and a composer, too."

- * Primary search in this case returns a few passages containing the correct answer, "Robert Schumann". But none of these passages are ranked high by our passage-scoring algorithms. One of these passage is:

"Clara Wieck Schumann: a German musician, one of the leading pianists of the Romantic era, as well as a composer, and wife of composer Robert Schumann."

- In this case, our scorers need to understand that 'married' in the Q matches 'wife' in the passage. This can be achieved by semantic relation detection but we do not have coverage for all semantic relations in Jeopardy
- * On the other hand, SER returns the following passage among others:
 - "Although Robert Schumann made some 'symphonic attempts' in the autumn of 1840, soon after he married his beloved Clara Wieck, he did not compose the symphony until early 1841."
 - This passage does not contain many of the original query terms such as "German", "romantic", or "pianist", and so was highly ranked only after we included "Robert Schumann" in the query. Since it uses the verb "married" with subject "Robert Schumann" (after applying some anaphora resolution) and object "Clara Wieck", it can be scored well by Logical Form Scorer using syntactic relations alone

• Syntactic-Semantic Graph

- 2 of the scorers operate on structural characteristics of Question and passages, i.e. syntactic-semantic graphs, i.e. parses of the Question & Passages. The Graph consists of:
 - * Nodes: terms in the Question/Passage
 - * Edges: encode syntactic (e.g. subject) and/or semantic relations (e.g. authorOf)
 - * Syntactic portion is derived by PAS
 - * Semantic structure derived by semantic relation detectors
- **Motivation:** Difference in text or order of text in the Question compared to that in the passage. But syntactic/semantic trees can get around this.
- **Deep Semantics:**
 - * Sometimes syntactic parse is not able to find the appropriate similarity between Question & Passage due to different ways in which the two are expressed grammatically.
 - * In such cases, deep semantic relation detectors are very useful. These detectors detect very specific relations, e.g. 'authorOf' which are represented as labeled edges in the syntactic-semantic graph
 - * Example: Consider the following Q & Passage:

"Who authored 'The Good Earth'?"

"Pearl Buck, author of 'The Good Earth'"

The two don't have similar syntactic parse as the first has main verb 'authored' while the second doesn't have a verb.

- On the hand, semantic relation can detect that there exists a 'authorOf relations between (who, 'The Good Earth') in the Q as well as b/w (Pearl Buck, 'The Good Earth') in the passage. This 'authorOf relation would be presented as an edge in the semantic graph connecting 'who'/'Pearl Buck' with 'The Good Earth'
- * Deep semantic relations connect strongly typed ontological entities, via ontological relationships (e.g., those between persons and works of art, exemplified by authorOf)
- * However, the coverage of our semantic relations detector is very low as it requires hand-crafted rules.

– Shallow Semantics

- * Shallow (unnamed) semantic relations between 2 or more entities can be very often detected using a variety of surface forms using a syntactic tree.
- * Examples:
 - Noun-Noun Modification vs. linking prepositional phrase
 - 'this Wyoming Senator' will match 'a senator from Wyoming' as we detect the same relation for both forms, i.e. (senator, relatedTo, Wyoming)
 - Thus, shallow semantic analysis abstracts away the syntactic differences between the noun compounding and noun-prepositional phrase post-modification.
- * The shallow semantics inventory (derived from an intensive data analysis) thus systematically encodes numerous patterns to detect alternative surface formulations of relations such as *relatedTo*, *instanceOf*, *subtypeOf*, *sameAs*, *coIndex*, *tLink*, and *gpAssociation*, i.e., the last two referring to temporal links (typically between a date and an event), and geopolitical association expressions such as "Alaska's capital" and "this Alaskan capital".

• Passage Scoring Algorithms

- **Passage Term Matching Scorer:** score based on the number of terms a passage contains from the set of question terms.
 - * Let $T = \{t_1, t_2, \dots, t_3\}$ be the set of terms extracted from the Q where a term may be a single word token, multiple-words (from a lexicon of common multiple-word tokens) token or a proper name.
 - * Score for passage i-th passage P_i is:

$$s_i = \frac{\sum_{t \in T, t \in P_i} \text{idf}(t)}{\sum_{t' \in T} \text{idf}(t')}$$

– Skip-Bigram Scorer:

- * Here the score is computed on the basis of matching **Skip-Bigrams** in the syntactic-semantic graphs of Question and Passage. A skip-bigram is a pair of terms linked directly in the graph or indirectly through only one intermediate node.
- * like Passage Term Match, Skip-Bigram score is largely a measure of co-occurrence; it determines the extent to which the answer appears in a passage with terms from the Question.

- However, Skip-Bigram specifically focuses on whether those words are close to each other in the syntactic-semantic graph

* **Example:**

"Who invented the motor driven phonograph?,"

- Skip-Bigram will give credit to passages that discuss "motor driven phonograph," "phonograph driven by a motor," "motor driving a phonograph," etc.
- Skip-Bigram will not give credit for passages that mention motors, driving, and phonographs in separate sentences or clauses and will thus ignore some passages that Passage Term Match awards a high score to.

- **Textual Alignment Candidate Scorer (TACS):** Here the order of matching terms is important. The score is computed by comparing terms and their order in the passage to those of the question, with the focus replaced by the candidate answer.

- * **motivated** by the fact that sometimes an answer which is very similar to the question is found, e.g.

"Who is the president of France?"

"Nikolas Sarkozy is the president of France."

- * TACS also needs to be able to handle partial matches

- Example

"In 1698, this comet discoverer took a ship called the Paramour Pink on the first purely scientific sea voyage."

should have a good match with:

"Edmund Halley made probably the first primarily scientific voyage to study the variation of the magnetic compass."

- The passage justifies only the part of the question about "first purely scientific sea voyage" without any support for "a ship named Paramour Pink".

- * A dynamic programming based approach (similar to edit distance) is used.

- Note that before we do the alignment, we replace Candidate Answer by the stub "CANDIDATE" in the passage and focus by the stub "FOCUS" in the Question and consider "FOCUS" == "CANDIDATE"

- * **Algorithm:** finds local alignments, i.e., optimum sub-sequences of the input passage P and question Q that align.

$$i', j' = \operatorname{argmax}_{i=1..|P|, j=1..|Q|} \operatorname{similarity}(P[1, \dots, i], Q[1, \dots, j])$$

- **Logical Form Answer Candidate Scorer (LFACS):**

- * LFACS attempts to align a syntactic-semantic graph of the content of a question to a syntactic-semantic graph of the content of a passage

- given the constraint that the node for the **focus** of the question must correspond to the node for the **CA** in the passage.

* Example:

The question, "Who wrote 'The Hobbit'?", maps to a graph with 2 nodes, "Who" and "The Hobbit", and 2 edges, (write-subject-who) and (write-object-The Hobbit)

The corresponding candidate passage, "Tolkien wrote 'The Hobbit'", also maps to a graph with 2 nodes, "Tolkien" and "The Hobbit" and 2 edges, (write-subject-Tolkien) and (write-object-'The Hobbit')

* There are cases where LFACS is able to provide a value that Textual Alignment cannot because the graphs encode a deeper analysis of the question and the passage than mere co-occurrence or proximity.

· Example:

"Dan Brown wrote several books and has read The Hobbit."

Tolkien, an English author born in the late nineteenth century, wrote The Hobbit."

From the perspective of simple text matching, the former is at least as good a match as the latter for "Who wrote The Hobbit?" However, syntactically (and semantically), the latter passage is a better match.

* The discriminating power that LFACS provides comes at the cost of substantial brittleness. Often, a passage retrieved for some candidate answer has many of the same words in roughly the same order as the question but does not relate those words using the same grammatical relationships.

· Example:

"Tolkien wrote several books and The Hobbit is the one that is easiest to read."

· This example should be interpreted as supporting the answer, but syntactically, it does not match the clue any better than the Dan Brown example. In some cases, this discrepancy can be addressed using semantic relation detection, as described earlier. However, relation detectors have limited coverage

· Score computation algorithm:

$$score(Psg) = \sum_{t \in Psg} idf(t) \times degreeOfMatch(t)$$

where,

$$degreeOfMatch(t) = termMatchScore(t) \times graphMatchScore(t)$$

where *termMatchScore* computes the extent to which a passage term, e.g. 'Dole', matches a question term, e.g. 'Bob Dole'.

$$graphMatchScore(t) = \max_{P \in paths} \prod_{t' \in P} termMatchScore(t') \prod_{e \in P} edgeMatchScore(e)$$

where *paths* is the set of paths from a question term *t* to the question focus. Each path *P* contains some intermediate nodes *t'* and edges *e*

- **Merging across passages:** There are 3 strategies for merging scores across passages with the same CA:
 - Maximum
 - Sum
 - Decaying Summer
 - Merging by maximum works best for LFACS while merging by Sum works best for Skip-Bigram and decaying sum works best for the rest.
- **Results:** All passage scoring with SER on full configuration Watson improved the accuracy by 3.3%
 - without SER the gain (again on full configuration) was only 1.3%
- **Papers**
 - Textual evidence gathering and analysis - IBM - 2012 - c54
 - * Anotated Copy
 - Shallow semantic ideas parsing taken from:
 - * COGEX: A Logic Prover for Question Answering - Moldovan et al - 2003 - c190
 - The algorithm for aligning graphs in LFACS
 - * Structure Mapping for Jeopardy! Clues - IBM DeepQA - 2011

10. Papers

- Building Watson: An Overview of the DeepQA Project - Ferruci - 2010 - c648
 - Anotated Copy
- Relation extraction and scoring in DeepQA - IBM - 2012
 - Anotated Copy
- Structured data and inference in DeepQA - IBM - 2012 - C53
 - Anotated Copy
- Special Questions and techniques - IBM - 2012
 - Anotated Copy
- Identifying implicit relationships - IBM - 2012
 - Anotated Copy
- Fact-based question decomposition in DeepQA - IBM - 2012
 - Anotated Copy

- Making Watson fast - IBM - 2012
 - Anotated Copy
- A framework for merging and ranking of answers in DeepQA - 2012
 - Anotated Copy
- When Did that Happen? – Linking Events and Relations to Timestamps - IBM - 2012
 - Anotated Copy
- UIMA: an architectural approach to unstructured information processing in the corporate research environment - Ferruci - 2004 - c747
- Semantic Technologies in IBM Watson - IBM - 2013
 - Anotated Copy
- Distant Supervision for Relation Extraction with an Incomplete Knowledge Base - IBM - 2013 - c112
 - Anotated Copy
- Parallel and Nested Decomposition for Factoid Questions - 2012
 - Anotated Copy
- Watson: Beyond Jeopardy! - IBM - 2013
 - Anotated Copy
- Unsupervised Entity-Relation Analysis in IBM Watson - IBM - 2015

1.10 Chained Reasoning Questions

1.10.1 Important Papers

1. Towards AI-Complete QA
 - Towards AI-Complete QA
 - **Dataset:** BabI (introduced by this paper)
2. Memory Networks
 - Memory Networks
 - End-to-End Memory Networks
 - Recurrent Entity Networks
3. Dynamic Memory Networks
 - Ask Me Anything: Dynamic Memory Networks for Natural Language Processing - Socher et al - ICML 2016 - c149
 - slides
4. Reasoning in Vector Space: An Exploratory Study of Question Answering
 - Reasoning in Vector Space: An Exploratory Study of Question Answering - Lee & Smolensky - 2015
5. RelNet: End-to-end Modeling of Entities&Relations RelNet: End-to-end Modeling of Entities&Relations - Bansal & McCallum - 2017

1.10.2 DataSets

1.11 Semantic Parsing

1.11.1 Overview & Important Papers

- Semantic Parsing is the process of mapping NL into a logical form (e.g. SQL) which is used to query a structured KB
- **Pros**
 - LF instead of answer makes the system robust to changes
 - Answer is independent of Q & Parsing mechanism
- **Cons**
 - Constrained by KB
 - Training Data Hard to come by
- KBs
 - Typically as triple stores using (Relation, Entity1, Entity2) format
- Solved using sequence-to-sequence model very similar to MT
 - use sequence model for encoding
 - Decode using standard MT mechanism
 - **Cons**
 - * Lack of training data
 - * Target side is highly complex
 - * How do you deal with proper nouns & numbers
- Solution to Data Sparsity
 - Avoid logical forms
 - exploit other forms of data, e.g. QA pairs or Question Paraphrases
 - **Papers:**
 - * Semantic Parsing on Freebase from Question-Answer Pairs - J. Berant et. al - c272 - EMNLP 2013
 - Local Copy
 - * Semantic Parsing via Paraphrasing - J. Berant et. al- ACL 2014 - c215
 - * Large-scale Semantic Parsing without Question-Answer Pairs - Reddy - c77 - 2014
 - * Semantic Parsing with Combinatory Categorical Grammars - Artzi - 2013
 - * Improving Semantic Parsing via Answer Type Inference - Yavuz et al - EMNLP 2016

· Results: $F1 = 51.6$ on WebQuestions

- Other mechanisms taken from MT
 - Attention
 - * Language to Logical Form with Neural Attention - Dong & Lapata - 2016
 - Exploit complex target side to constrain generation
 - * Latent Predictor Networks for Code Generation - DeepMind - 2016
 - * Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision - Google / NWU 2016
 - Semi-supervised Training to counter data sparsity
 - * Semantic Parsing with Semi-Supervised Sequential Autoencoders - Deepmind - EMNLP 2016
- Generation of Answers using Multiple Sources (KBs)
 - Latent Predictor Networks for Code Generation - DeepMind - 2016
 - Pointer Networks - GoogleBrain - NIPS 2015 - c60
- Stanford Papers on Semantic Parsing
 - cs224u slides
 - Building a Semantic Parser Overnight - 2015
 - Bringing machine learning and compositional semantics together
 - Core concepts for semantic parsing - video
 - Semantic Parsing Models - Video
 - Learning to Map Sentences to Logical Form - 2005 - WashU - c457
- Classic Papers
 - Learning to Parse Database Queries - Zelle & Mooney - 1996 - c360
 - * Simple Commands : intent/context generation (e.g. in wit.ai)

1.11.2 DataSets

1. KBs

- Freebase (overriden by Google Knowledge Graph) - 1.9B Triples
- WikiData - 25M entities
- OpenStreetMap - 3B triples on Geography
- GeoQuery - 700 facts about USA Geography

2. Training Data

- Free917 - 917 freebase anotated Qs
- GeoQuery - 880 questions
- NLMaps - 2380 questions

1.12 Natural Language Inference

- Improved Semantic Rep From Tree-Structured LSTM Networks - 2015 - 229c
- A Decomposable Attention Model for Natural Language Inference - 2016
- Recursive Deep Models for Semantic Comp. Over a Sentiment TB - 2013 - 1123c
- Reasoning about Entailment with Neural Attention - 2015
 - **Goal:** Given a pair of sentences, denoted premise and hypothesis, to determine if the hypothesis is entailed/contradicted by the premise. This is a useful task for many NLP applications like Relation Extraction and Question & Answering
 - **Paper Overview:** The idea here is to use RNN (with LSTMs) with attention to do fine grained reasoning across words, phrases and larger constructs across a pair of sentences.
 - **Data:** Stanford Natural Language Inference Corpus consisting of 570K pairs of sentences (created by human annotaters)
 - **Results on Our Implementation:**
 - * We achieved results which were 3 percentage point below the results claimed in the paper. But this was because we did not put any effort in fine tune hyper-parameters.
- A large annotated corpus for learning natural language inference - 2015
- Excitement Open Platform
- Learning to recognize features of valid textual entailments - 2006 - 143cooperation
- Recognising Textual Entailment with Logical Inference - 2005 - 185c
- <http://nlp.stanford.edu/projects/snli/>
- cs224u slides
- Bowman cs224u guest lecture slides

1.13 Pipeline

1.13.1 AskMSR

- Web question answering: is more always better - Microsoft Research - 2002 - c370]
- An Analysis Of The AskMSR Question-Answering System - Microsoft Research - 2002 - c229
- AskMSR: Question Answering Using the Worldwide Web - Microsoft Research - 2002 - c50

1.13.2 Ephyra

- A pattern learning approach to question answering within the ephyra - Karlsruhe - 2006

1.13.3 Jacana

- Answer Extraction as Sequence Tagging with Tree Edit Distance - John Hopkins - 2013 - c77
- Automatic Coupling of Answer Extraction and Information Retrieval - John Hopkins ACL 2013

1.13.4 openQA

- Open source question answering framework.

1.13.5 OAQA

- <https://oaqa.github.io/>
- Towards the open advancement of question answer systems - Ferruci - 2009

1.13.6 WatsonSim

- Watsonsim: Overview of a question answering engine - UNCC - 2014

1.13.7 YodaQA

- YodaQA: A Modular Question Answering System Pipeline - Petr Baudis - 2014

1.13.8 QuASE

- Open Domain Question Answering via Semantic Enrichment - Sun et al - 2015

2 Paraphrase Detection

2.1 Taxonomy

- Squibs: What Is a Paraphrase? - Bhagat & Hovy - 2013

2.2 Dataset Papers and Datasets

2.2.1 Microsoft Research Paraphrase Corpus

- Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources - Dolan et al. - COLING 2004 - c469
- Automatically Constructing a Corpus of Sentential Paraphrases - Dolan et al - Microsoft Research - 2005 - c107
- SVM for Paraphrase Identification and Corpus Construction - Brockett & Dolan - Microsoft Research - 2005 - c54

2.2.2 Other DataSet Papers

- A Continuously Growing Dataset of Sentential Paraphrases - Lan et al - UMD/UPENN
- PPDB: The Paraphrase Database - Ganitkevitch et al - John Hopkins - HLT NACCL 2013
 - local copy

2.2.3 Datasets

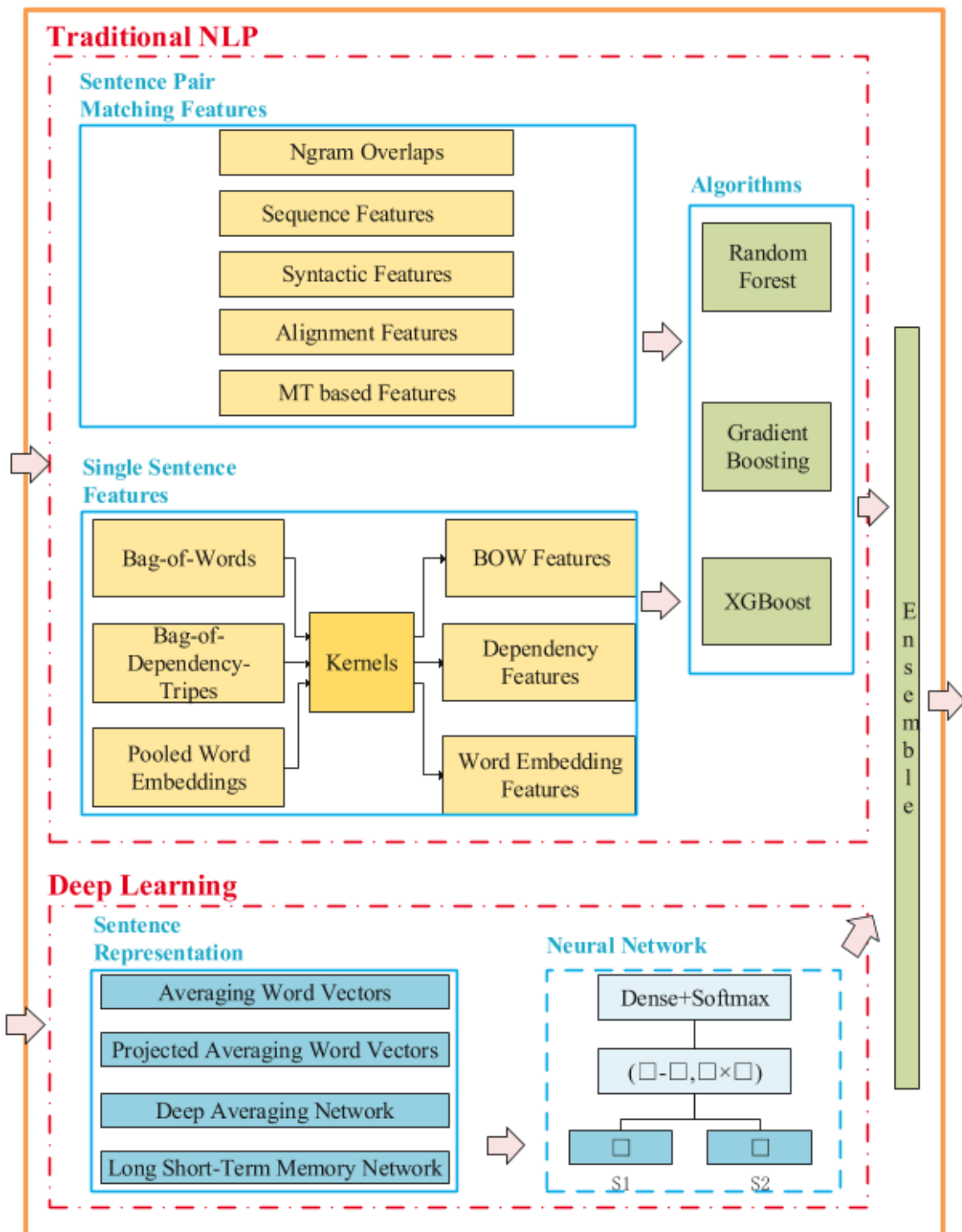
- WikiAnswers Paraphrase Dataset
 - 18 million Q paraphrase pairs (2.5 million distinct Qs)
 - <http://knowitall.cs.washington.edu/paralex/wikianswers-paraphrases-1.0.tar.gz>
- Quora Question Paraphrase Data

- 400K Q paraphrase pairs
- <https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs>
- Microsoft Research Paraphrase Corpus
 - 5800 sentence paraphrase pairs
 - <https://www.microsoft.com/en-us/download/confirmation.aspx?id=52398>
- SemEval STS (Semantic Task Similarity) data
 - 14000 paraphrase pairs across 2012-2015 STS tasks
 - http://ixa2.si.ehu.es/stswiki/index.php/Main_Page
- Sick Dataset
 - <http://clic.cimec.unitn.it/composes/materials/SICK.zip>
- AskUbuntu Data
 - <https://drive.google.com/drive/u/0/folders/0B-btHzfJjPnobXZ0MndjSkxkRkk>

2.3 Important Papers

2.3.1 ECNU at SemEval 2017

- Overview:
 - A giant ensemble of traditional NLP features as well as DNN features. The arch is shown below



– **Traditional NLP features:** There are 2 sub-types of Traditional NLP features

* **Sentence-pair features:** Features capturing interaction b/w within the sentence pair. These include

- N-gram overlaps:
- at word level (lemmatized and not)
- at character level
- Sequence features

- longest common prefix/suffix/substring/sequence
- levenshtein distance
- Syntactic Parse features:
 - uses tree kernels to calculate similarity b/w 2 syntactic parse trees
- Alignment features (as in Sultan et al)
- MT based features (as in Zhae et al, 2015)

* **Single sentence features**

- bag of word vectors weighted by IDF
- Bag-of-triples of a dependency tree, where a triple is (dependency-label, governor, subordinate) tuple in a dependency tree
- Word Embedding Features
 - using 4 different pre-trained embeddings (word2vec, glove 100d/330d & paragram)
- A set of kernel functions are used to compute similarity of the 3 kinds of vectors above

– **DNN features**

- * A sentence is represented by combining paragram vector of its words in various ways:

- averaging
- averaging followed by projecting
- DAN
- LSTM

- * to obtain the vector of sentence pair, given two single sentence vectors, we first use a element-wise subtraction and a multiplication and then concatenate the two values as the final vector of sentence pair representation

- The resulting vector is passed thru fully connected hidden layer and output layer to predict the similarity as a probability value using softmax function

– Ensemble: Simple Averaging

- Dataset: SemEval 2016/2017 STS data
- Results: Best at SemEval 2017
- ECNU at SemEval-2017 Task 1: Winner of Cross-lingual Semantic Textual Similarity

– local copy

2.3.2 Discriminative Improvements to Distributional Sentence Similarity

- **Overview:**

- Latent vector space representation computed using NMF
- Starts with a co-occurrence matrix A whose rows are sentences and whose columns are features (like unigram/bigram counts or dependency pairs)
- **Main Idea:** Instead of throwing away the label information while doing matrix factorization, use it for identifying discriminative features, i.e. features which are much more likely to co-occur in the paraphrase than in the non-paraphrase case or vice-versa.

- * use KL-divergence between the probability of a feature co-occurring during a paraphrase and the probability of a feature co-occurring in a non-paraphrase case to re-weight the co-occurrence matrix

- * The weight is called TF-KLd because the TF (co-occurrence count) is re-weighted by KL divergence

- Let

$$p_k = P(w_{ik}^{(1)} | w_{ik}^{(2)} = 1, l_i = 1)$$

probability that sentence $w_i^{(1)}$ contains feature k , given that k appears in $w_i^{(2)}$ and the two sentences are labeled as paraphrases, i.e. $l_i = 1$

- Let

$$q_k = P(w_{ik}^{(1)} | w_{ik}^{(2)} = 0, l_i = 0)$$

probability that sentence $w_i^{(1)}$ contains feature k , given that k appears in $w_i^{(2)}$ and the two sentences are labeled as not paraphrases, i.e. $l_i = 0$

- The KL divergence:

$$KL(p_k) = \sum_x p_k(x) \log \frac{p_k(x)}{q_k(x)}$$

is then a measure of the discriminability of feature

- Latent vectors v_1 and v_2 of a sentence pair is converted into a feature vector as follows:

$$[v_1 + v_2, |v_1 - v_2|]$$

where $[,]$ denotes concatenation

- Instead of using similarity or distance between the latent representations of a pair of sentences, directly use the dimensions of the latent vectors (or a vector computed using them as above) as features for a classifier downstream.
- Combine with surface features that capture fine-grained similarity b/w sentences, e.g. by counting unigram, bigram and dependency relation overlaps.

- **Data set:** MSRPC

- **Results:**

- on MSRPC: Acc = 80.4, F1 = 85.9

- Discriminative Improvements to Distributional Sentence Similarity - Ji & Eisenstein - EMNLP 2013

- local copy
- Results

2.3.3 MayoNLP at SemEval-2016

- Preprocessing:

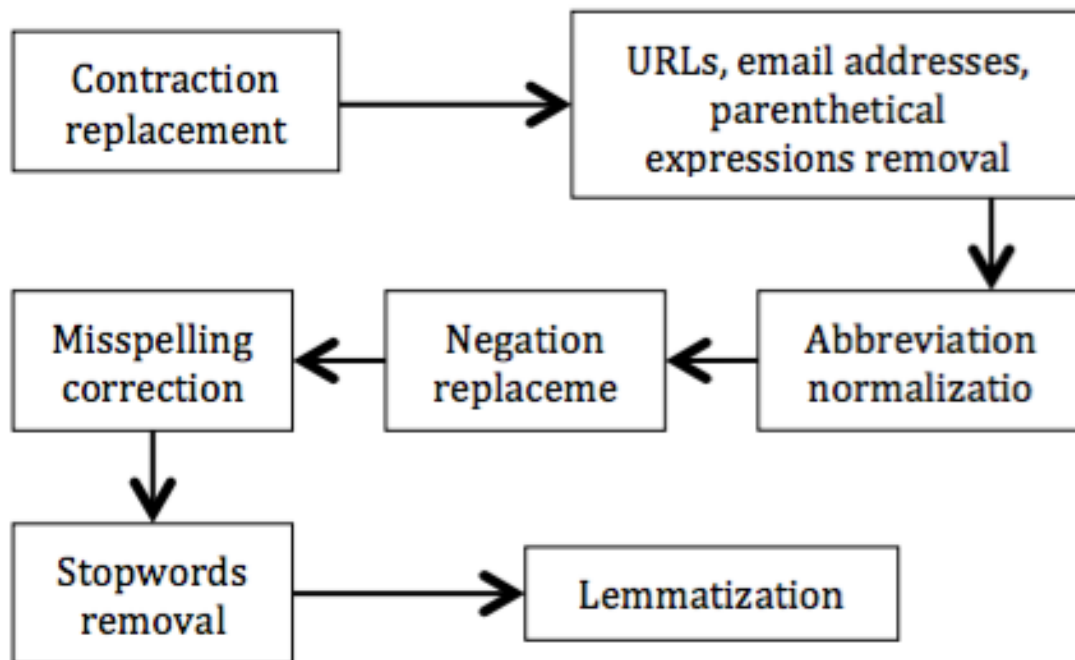


Figure 1: Workflow of preprocessing

- Model Architecture

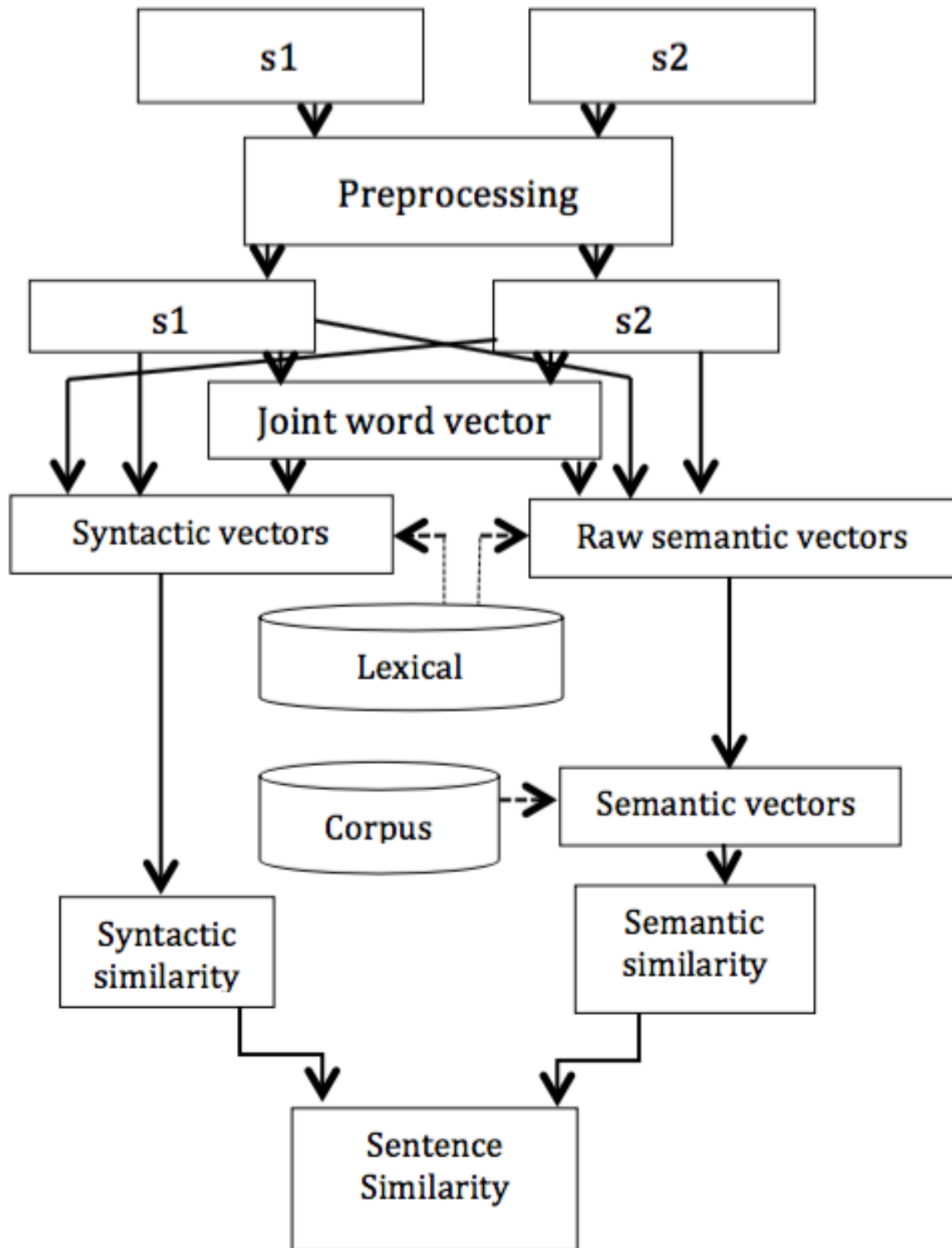


Figure 2: System architecture (Run 1)

- Uses 3 features for modeling the semantic similarity of a pair of sentences:
 - semantic similarity
 - syntactic similarity
 - semantic similarity computed using a DNN

- Computes semantic similarity b/w a pair of sentence by first converting each of them into a vector
 - First computes a joint word vector (JWV) as a set of unique words from the sentence pair
 - * Converts each sentence into a vector, where i-th element is the similarity score of some word in the sentence which is closest (as per WordNet semantic similarity) to the i-th word in JWV. The wordNet semantic similarity score is:

$$s(w_i, w_j) = e^{\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}}$$

- * where l is the length of the shortest path b/w w_i and w_j and h returns a measure of depth in WordNet.
 - * Each index of the vector is weighted by the IDF of the i-th word in JWV
 - * Semantic similarity b/w the sentence pair is the cosine similarity of their vectors
- Computes syntactic similarity b/w a pair of sentences by converting each sentence into a vector
 - Converts each sentence into a vector, where i-th element is the index of that word in the sentence which is closest (as per $s(w_i, w_j)$ above) to the i-th word JWV. So the vector captures the position of the most similar word in the sentence corresponding to the i-th word in JWV.
 - Syntactic similarity is computed as:

$$s_{syn} = 1 - \frac{|o_1 - o_2|}{|o_1 + o_2|}$$

- * where o_1 and o_2 are the syntactic vectors corresponding to the sentence pair
- **Word Hashing:** DNN just maps the bag of word vector for each sentence to a character n-gram vector (where $n = 3$), by decomposing each word into a series of n-grams. But the word is first padded with # character on each end. So a sentence is represented as a bag of character n-grams of its words (each element of the vector identifies whether a corresponding character n-gram is present or not)
 - The sentence vectors are then projected by a series of non-linear projections (using hidden layers and tanh) into a semantic vector space.
 - Then cosine similarity is used to compute the similarity of the semantic vectors. A linear projection is finally used to map the cosine similarity to a value b/w 1 and 5.
- MayoNLP at SemEval-2016 Task 1: Semantic Textual Similarity based on Lexical Semantic Net and Deep Learning Semantic Model - Afzal et al - Mayo Clinic - NAACL-HLT 2016
 - local copy
- **Dataset:** SemEval 2016/2017 STS
- **Results:** Best on question-question paraphrase

2.3.4 Neural Paraphrase Identification of Questions with Noisy Pretraining

- **Main Ideas:**

- Soft-alignment between pairs of questions where the alignment is done at the granularity of a context window.

- * Let $a = (a_1, \dots, a_{l_a})$ and $b = (b_1, \dots, b_{l_b})$ be a pair of input texts (e.g. Question pair). Each term a_i and b_j are encoded as a vector of dimension d .
- * A context window of size c around a word a_i is defined as:

$$\bar{a}_i = [a_{i-c}, a_i, \dots, a_{i+c}]$$

- * **Attend:** Each context window in one Q is matched against all the context windows in the other Q to compute a similarity score:

$$e_{ij} = F(\bar{a}_i)^T F(\bar{b}_j), \forall i, j$$

where F is a feed forward NN.

- For each context window \bar{a}_i in one Q, a subphrase β_i of the other Q is softly aligned to \bar{a}_i using similarity scores as weights

$$\beta_i = \sum_{j=1}^{l_b} \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})} \bar{b}_j$$
$$\alpha_i = \sum_{j=1}^{l_a} \frac{\exp(e_{ij})}{\sum_k \exp(e_{kj})} \bar{a}_i$$

- * **Compare** aligned phrases $\{\bar{a}_i, \beta_i\}_{i=1}^{l_a}$ and $\{\bar{b}_j, \alpha_j\}_{j=1}^{l_b}$ are separately compared using a feed forward network G :

$$v_{1,i} = G([\bar{a}_i, \beta_i]) \forall i \in [1, \dots, l_a]$$

$$v_{2,j} = G([\bar{b}_j, \alpha_j]) \forall j \in [1, \dots, l_b]$$

- * **Aggregate:** Finally, the sets $\{v_{1,i}\}_{i=1}^{l_a}$ and $\{v_{2,j}\}_{j=1}^{l_b}$ are aggregated and the result is predicted using another feed-forward NN

- Instead of word embeddings, use character n-gram embeddings
- Use large paraphrase data corpus for pre-training of all the stages of the model

- Neural Paraphrase Identification of Questions with Noisy Pretraining - Google - 2017

- Results on Quora: Acc = 88.4

2.3.5 BiMPM Model

- Main Ideas:

- capture interaction between a pair of text sequences at various levels of granularity using different perspectives
- compose word embeddings from character level embeddings by feeding embeddings of characters within a word to an LSTM

- *Architecture

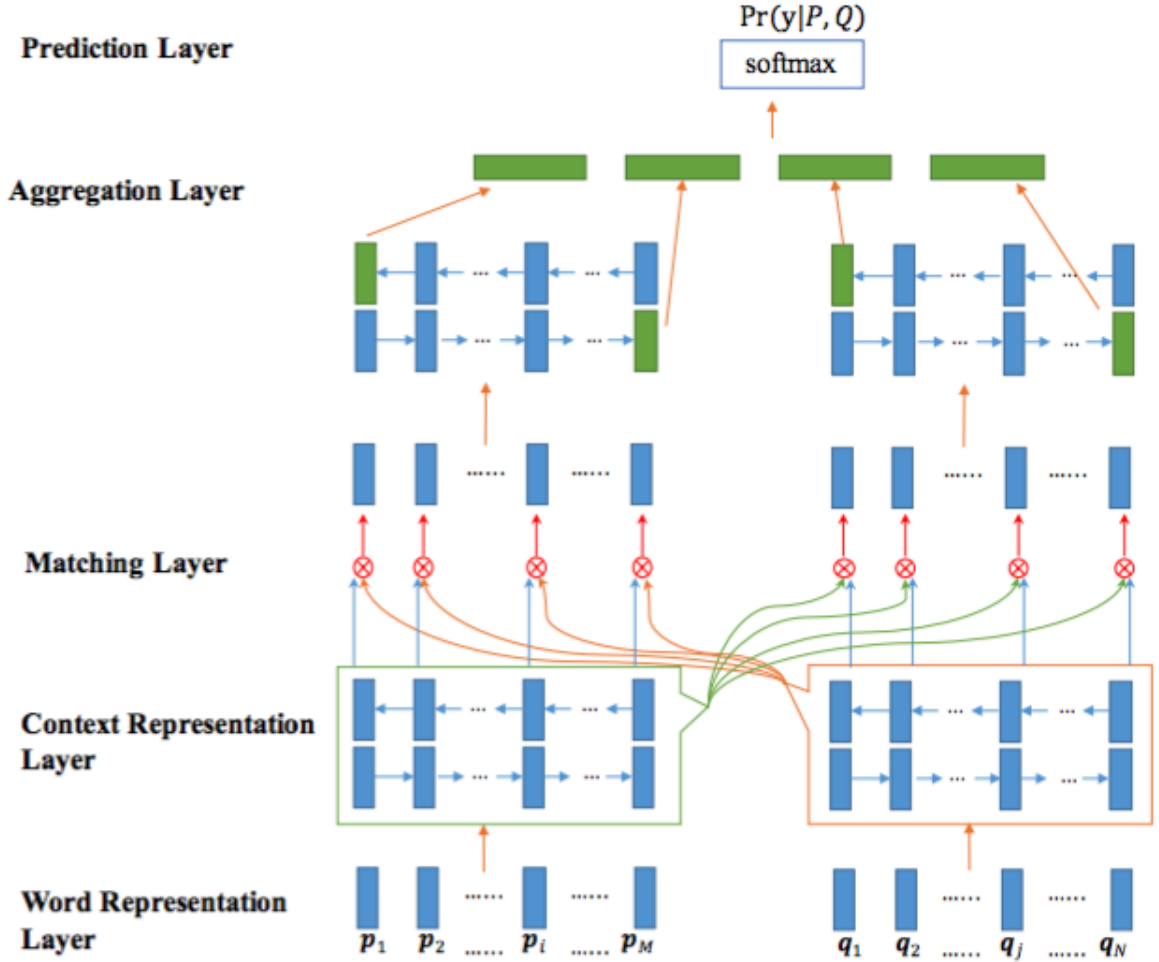


Figure 1: Architecture for Bilateral Multi-Perspective Matching (BiMPM) Model, where \otimes is the multi-perspective matching operation described in sub-section 3.2.

- **Word Representation Layer:** The input layer uses word embeddings from pre-trained word vectors (e.g. GloVe) as well as word embeddings composed from character level embeddings (using an LSTM)
- **Context Representation Layer:** Each time step (i.e. word) in a sequence is mapped to a context vector by feeding the sequence into a BiLSTM. The purpose of this layer is to incorporate contextual information into the representation of each time step of a sequence

* Instead of the above 2 layers, could we directly use a sentence encoding layer like InferSent ?

- **Matching Layer:** This is the main layer of the model. The goal of this layer is to compare each contextual embedding (time step) of first sequence with all contextual embeddings (time steps) of the second sequence (by aggregating context vectors across all time-steps of the 2nd seq in some way which may or may not be dependent on the contextual vector of the given time step in the 1st seq). This is done in both directions (i.e from first to second seq and vice-versa).

* A multi-perspective matching operation is defined for this purpose.

$$m = f_m(v_1, v_2; W)$$

where v_1 and v_2 are

d

-dimensional vectors and $W \in R^{l \times d}$ is a trainable parameter where l is the number of perspective and the returned value m is an

l

-dimensional vector. Each element m_k is a matching value from

k

-th perspective

$$m_k = \cos(W_k \cdot v_1, W_k \cdot v_2)$$

where \cdot is the element-wise multiplication and W_k is the

k

-th row of W which controls the

k

-th perspective and assigns different weights to different dimensions of the

d

-dimensional space.

- * 4 matching operations are defined which match one time step of one sequence with all time-steps of the other sequence as show below.

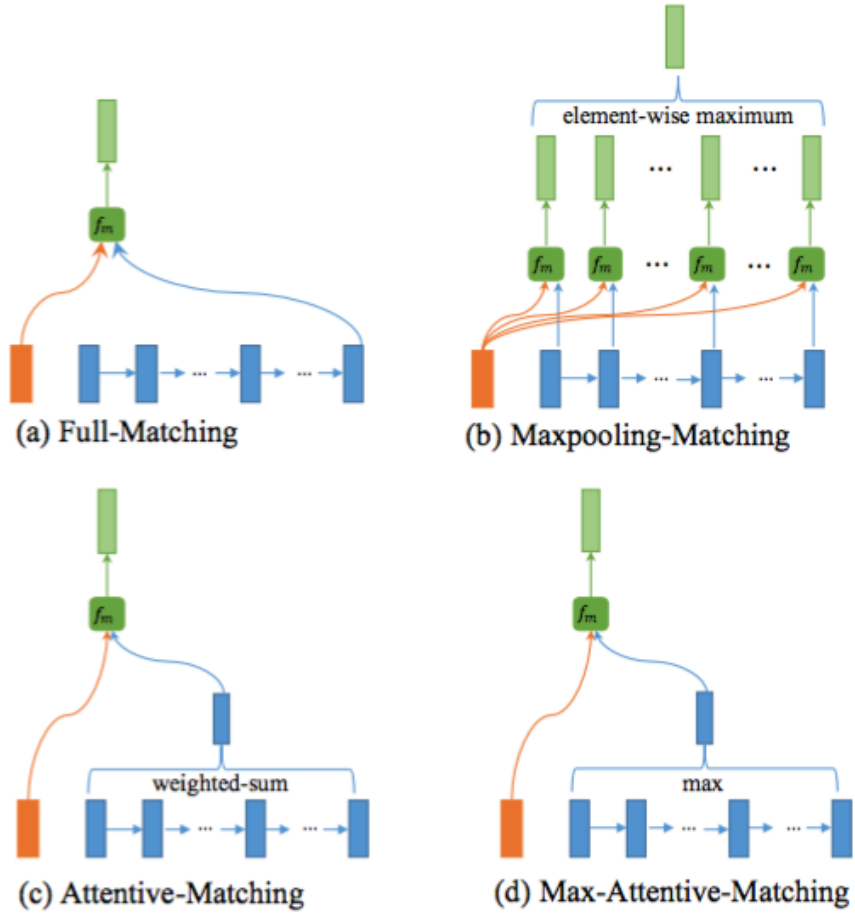


Figure 2: Diagrams for different matching strategies, where f_m is the multi-perspective cosine matching function in Eq.(3), the input includes one time step of one sentence (left orange block) and all the time-steps of the other sentence (right blue blocks), and the output is a vector of matching values (top green block) calculated by Eq.(3).

* This layer results in 8 fixed size vectors (4 matching strategies times 2 directions of the BiLSTM) for each time step in each sequence. These are concatenated into a single vector for each time-step.

- **Aggregation Layer:** This layer takes the resulting vector for each time step from the matching layer and passes thru another BiLSTM to produce 4 vectors (2 for each sequence on either end of their respective BiLSTMs). The BiLSTM is shared b/w the 2 sequences.
- **Prediction Layer:** The 4 fixed size vectors from the previous layer are concatenated and all their dimensions are used as features into a 2-layer feed-forward NN with softmax as the output layer.

- Bilateral Multi-Perspective Matching for Natural Language Sentences - Wang et al - IJCAI 2017 - c28

- local copy
- Results on Quora: Acc = 88.17
- Code

2.3.6 more papers

- Probabilistic tree edit models with structured latent variables for textual entailment and question answering - Wang & Manning - COLING 2010 - c85
- Molding CNNs for text: non-linear, non-consecutive convolutions - Lei et al - MIT CSAIL - EMNLP 2015
- Semi-supervised Question Retrieval with Gated Convolutions - Lei et al - MIT CSAIL - HLT NAACL 2016
- SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity - Agirre et al -NAACL-HTL 2012

2.4 State of Art Papers on MSRPC

- Syntax-Aware Multi-Sense Word Embeddings for Deep Compositional Models of Meaning - Cheng and Kart-saklis - EMNLP 2015
 - Results on MSRPC: Acc = 78.6%, F1 = 85.3%
- Sentence Similarity Learning by Lexical Decomposition and Composition - IBM research - COLING 2016
 - local copy
 - Results on MSRPC: Acc = 80.4, F1 = 84.7
- Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks - He et al - EMNLP 2015 - c57
 - Results on MSRPC: Acc = 78.6, F1 = 84.7
- Re-examining Machine Translation Metrics for Paraphrase Identification - Madnani et al - HTL-NAACL 2012 - c95
 - Results on MSRPC: Acc = 77.4, F1 = 84.1
- Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection - Stanford - c455 - 2011
 - Results on MSRPC: Acc = 76.8%, F1 = 83.6%
- Transfer Learning: From a Translation Model to a Dense Sentence Representation with Application to Paraphrase Detection - ferguson et al - 2017
 - Results on MSRPC: Acc = 76.3%, F1 = 83.2%
- Paraphrase identification as probabilistic quasi-synchronous recognition - Das et al - ACL 2009 - c195
 - Results on MSRPC: Acc = 76.06%, F1 = 82.5
- Using Dependency-Based Features to Take the “Para-farce” out of Paraphrase - Wan et al - 2006 - c72
 - Results on MSRPC: Acc = 75.6, F1 = 83
- A Semantic Similarity Approach to Paraphrase Detection - Fernando et al - 2008
 - Results on MSRPC: Acc = 74.1., F1 = 82.4
- Corpus-based and Knowledge-based Measures of Text Semantic Similarity - Mihalcea et al - AAAI 2006 - c792
 - Results on MSRPC: Acc = 81.3, F1 = 70.3

2.5 Review Paper

- Generating Phrasal and Sentential Paraphrases: A Survey of Data-Driven Methods - Madnani et al - Computational Linguistics - 2010 - c156

2.6 Related Papers

- Learning to Paraphrase for Question Answering - Dong & Reddy - 2017
 - 2nd Best results on QA in WebQuestions
- Semantic Parsing on Freebase from Question-Answer Pairs - J. Berant - c272 - 2013
 - Local Copy
- Semantic Parsing via Paraphrasing - 2014 - c215
- Paraphrase-Driven Learning for Open Question Answering- 2013 - c152 (aka PARALEX)
- Ask the Right Questions: Active Question Reformulation with Reinforcement Learning - Buck et. al - Google - 2017
- Paraphrasing Revisited with Neural Machine Translation - Lapata et al - EACL 2017

2.7 Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment

- Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment - Barzilay, Lee - HLT-NAACL 2003 - c460
 - local copy

2.8 Theory

- Approaches:
 - Alignment approaches
 - * n-gram features
 - * sequences/edit-distance
 - * MT features
 - * Dependency pairs
 - Vector Space approaches
 - * use
- Try using the word embeddings learned from snli data
 - Supervised Learning of Universal Sentence Representations from Natural Language Inference Data - Facebook AI - 2017
- Use phrasal alignment for features for paraphrasing as used in:

- Semantic Parsing via Paraphrasing - 2014 - c215
- 2 different approaches phrasal alignment:
 - * Based on phrase mapping learnt using paraphrase questions corpus (extracted from wikianswers)
 - the phrase alignment was done using MGIZA++ (an alignment tool used in MT)
 - * Based on phrase mappings in PPDB
 - PPDB phrase mappings were done by translating a pair of english phrases to another language and if the resulting translation was same then the english phrases were deemed to be paraphrases
- Can we use SRL for paraphrase
 - Example
 - * how do I remove paint from my hair?
 - * how do I remove paint from pipes?
 - * Even though the Q pair above have the same predicate "remove", they are quite different. SRL would catch that.
- Can we use concepts from IE, which maps phrases (or sub-sequence of words in a question) to canonical entities and relations
 - then compare entities/reactions across a pair of sentences to derive features for paraphrasing
- Can we use semantic parsing as an intermediate step in paraphrase detection
 - use derived logical forms as latent features
 - * inspired by - Semantic Parsing via Paraphrasing - Berant et al. - 2014

3 Question Generation

3.1 Rule based approaches

3.1.1 Syntax Based

- **Reference:**
 - Good Question! Statistical Ranking for Question Generation - Heilman & Smith - HLT-NAACL 2010 - c90
 - local copy
- **Introduction:**
 - **Over-generate and Rank:**
 - * Uses manual rules to perform a sequence of syntactic transformations (subject-auxiliary inversion) to turn declarative sentences into Qs.

- * Qs are then ranked using LR.
- Restricted to factual information in texts

- **Glossary:**

- **source sentence:**
- **answer phrase:**
- **question:**

- **Rule-based Over-generation:**

- **Sentence Simplification:** For a given sentence, a set of declarative sentences is extracted

- * This requires removing

- appositives
- leading conjunctions
- sentence-level modifying phrases

- * declarative sentences can also be extracted from subordinate clause or noun phrase or prepositional phrase

- * Thus multiple declarative sentences can be derived from a single source sentence

- * Implemented using tregex (a tree query lang) for matching patterns and tsurgeon (a tree manipulation lang built on top of tregex) for deleting, moving, inserting and updating nodes

- **Question Transformation**

- * **Answer Phrase Selection:**

- Answer phrase could be a noun phrase or a prepositional phrase
- These enable *who*, *what*, *where*, *when*, *how much* Qs

- **Identifying Unmovable Phrases in a Sentence**

- In English, various constraints determine whether phrases can be involved in WH-movement and other phenomena involving long distance dependencies.
- noun phrases are “islands” to movement, meaning that constituents dominated by a noun phrase typically cannot undergo WH-movement. Thus, from "John liked the book that I gave him", we generate "What did John like?" but not "Who did John like the book that gave him?"

- * **Generating Question Phrases:** Involves the following steps as depicted in the picture below

- **Remove Selected Answer Phrase:** Iteratively remove each possible answer phrase from the simplified sentence
- **Main Verb Decomposition:** see picture below
- **Subject-Auxiliary Inversion:** See picture below

- **Movement and Insertion of Question Phrase:** The system annotates the source sentence with a set of entity types and then uses these entity labels along with the syntactic structure of a given answer phrase to generate zero or more question phrases

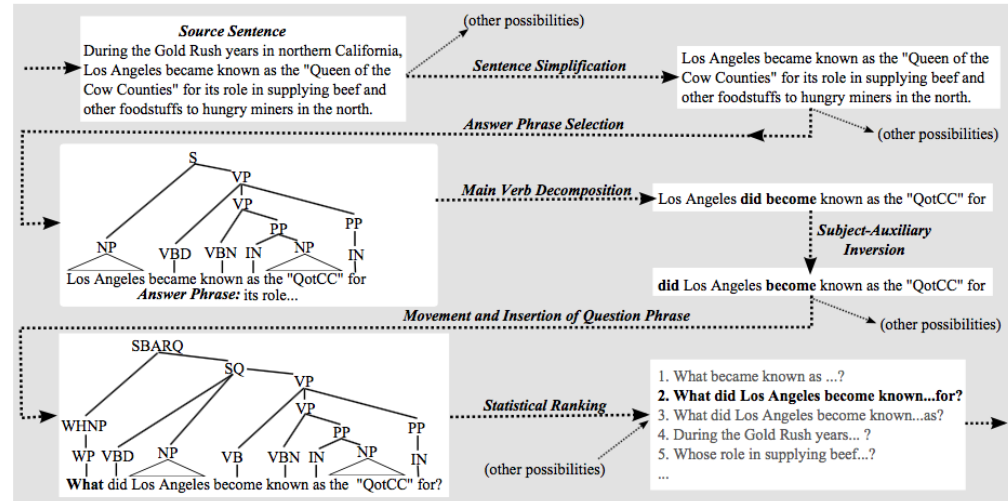


Figure 1: An illustration of the sequence of steps for generating questions. For clarity, trees are not shown for all steps. Also, while many questions may be generated from a single source sentence, only one path is shown.

- * **Multiple Qs for each Source Sentence:** the above process results in multiple Qs per source sentences as there are multiple declarative sentences per source sentence as well multiple answer phrases per declarative sentence and each answer phrase can generate multiple Qs.

• Rating Qs for eval

- **Human judged:** All Qs generated by the above 2 steps are judged by humans for acceptability. A Question is acceptable if it does not match any of the criteria below for not accepting a Q.

– Criteria for not accepting a Q:

- * Ungrammatical
- * Does Not Make Sense
- * Vague
- * Formatting error
- * Wrong WH-word
- * Missing Answer
- * Obvious Answer
- * Other error
- * None: i.e. acceptable

• Learning to Rank Qs

- **Dataset:** Qs generated from Wikipedia/Newswire using the above 2 steps
- **Features**
 - * Length

- * WH-words
- * Negation
- * Uni and tri-gram language model features
- * Grammatical
 - count of nouns, pronouns, adj, verb, conj, prep etc ?
- * Transformation Type(s)
- * Vagueness
- * Histograms of length and counts of grammatical objects

• Results

- **Without Ranking:** 27.7 % were acceptable across all Qs generated by this approach
- **After Ranking:** Improves to 52% when only 20% ranked documents are considered.
 - * Grammatical features are the most important
 - * All feature contribute significantly to the prediction.

3.1.2 Semantic based

1. UPENN Research

- Question Generation from Paragraphs at UPenn: QGSTEC System Description - Mannem et al - 2010 - c60
 - Local Copy
- Uses Semantic Role Labeling (SRL) to identify arguments of a predicate and their roles
 - The mandatory arguments (ARG0 to ARG5) as well as optional argument ARGM-{\LOC|TMP|CAU|..} are targets (i.e. answers) for questions generation
 - Also args for linking verbs (computed by dependency parsing) are also targets
 - First sentence of a paragraph is a target for a general question
- Each target in the context of its parse tree is transformed into a Question
 - The target itself is deleted from the sentence and if it has a preposition than the target is replaced by the preposition
 - Wh-type (i.e. who/where/why/what/when/how) is determined based on the argument role and the entity type of the target
 - * Default is what
 - Next, all the optional arguments to the left of the predicate are moved to the end of the sentence.
 - A verb complex of the predicate (i.e. the auxiliary and modal verbs around the predicate) is determined using dependency tree.

- * This is used for WH-inversion, i.e. the auxiliary is added to the beginning of the sentence and WH is added before that.
- * If the predicate has no auxiliary then depending on the POS of the predicate, did/do/does/ is added as the auxiliary and then WH-type is added before that
- Finally the lemma of the predicate is added to the end of the sentence
- For the case of the first sentence of the paragraph:
 - * if that sentence has is a linking verb, then the arg to the right of the predicate is a good target
 - * Otherwise, Q is formed by relativizing the right argument of the main clause of the sentence.
- Finally the resulting Qs are ranked
 - A large number of questions are generated, one for each target. They are ranked based on
 - * The level of the predicate in the dependency tree - higher the level, the more important the generate Q is
 - * Q with pronouns are less preferred as the system has no coreference resolution.

2. CMU research on generating Qs from informational and narrative text

(a) Understanding Mental States in Natural Language

• Introduction

- Natural language (NL) carry mental state or worldview of different characters
- To understand (NL) we need to model the mental state of worldview of these characters
- **mental state:** belief, supposition, intent, perceived reality, advice, states of knowledge
 - * also situations expressed by tenses (past, present, future)
- This paper has 2 goals:
 - * Extracting mental states from text using linguistic patterns of mental states
 - * Representing extracted mental state in a form/model which eases inference on it
- **Mental State Extraction:**
 - * Mental state is extracted from text using (agent, psych-term) pairs
 - * Examples:
 - (1) a. The police believe the thieves were trying to steal a solar panel from Sarah's tin roof.
 - In this example, "the police believe" is the (agent, psych-term) pair
 - (1) b. She (little red-cap) was surprised to find the cottage-door standing open.
 - In this example, "little red-cap was surprised" is the (agent, psych-term) pair

*

- **Mental States** are modeled formally using **mental context**

- * See the next paper for a detailed description of *mental coninstant

- **Mental Context:**

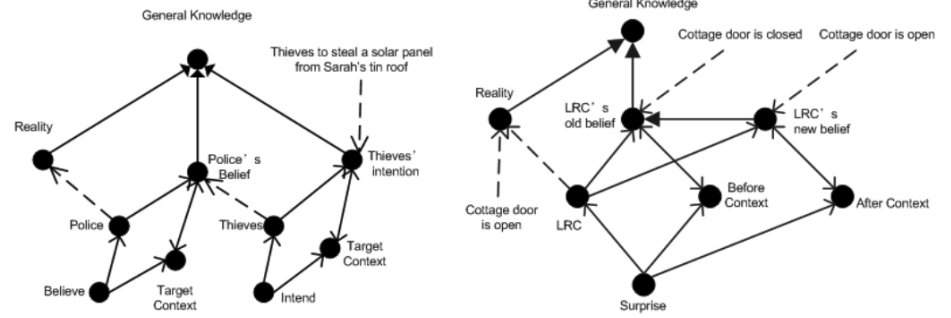
- Construct for modeling **mental state** or worldview of a character/person
- Inference is done with respect to a mental context (i.e. wrt a person/characters point of view)
- A context is a container that holds a set of descriptions. It can either be used to represent the “mental state” or a “situation”
 - * A description is an action (predicate/sense) frame with appropriate slots for roles like agent, patient etc. filled in.
 - For example believe/try are actions with appropriate slots, i.e. the 'believer' or the 'person trying' and the 'belief' or the 'intent' (of trying)
- Context is like a scope in programming language. Once you activate a context, you can reason with information within the context without worrying about information in other contexts
 - * When a context is activated, all its ancestor contexts (with their nodes and links) in the hierarchy are also activated
- Contexts can be nested allowing us to model one person’s belief of someone elses’ belief.
- We can also model changes in mental state over time. We store one copy of the mental context for each point in time.
 - * Evolution of context from one time slice to next happens by cloning the contents of the original context and make appropriate changes (additions, deletions, updates) to it.
- We can also model active vs. dormant memories using contexts.
 - * Dormant memories exist in the mind, temporarily inactive, but can be reactivated by entities or events
- In our case, psych-terms such as “realize” will be matched to Scone psych-events, which connect a set of mental contexts.

- **System Overview:**

- A parser combines several natural language processing components to extract useful information from raw text. The information is then integrated and filled into psych-term-argument templates to generate an intermediate semantic form called **mental operation**.
- The inference engine translates mental operations into a situation model represented by a semantic network (using **mental contexts** as a building block).

– **Example Output:**

- * Figure 2(a) and 2(b) show the semantic networks generated by the system given input sentences (1a) and (1b), respectively. As shown in the figure, the semantic representation of mental states is a set of **mental contexts** attached to different characters.



(a) The semantic representation of (1a). (b) The semantic representation of (1b).

- * Arrows with big solid heads represent “sub-context” relation. Arrows with dotted lines represent “in-context” relation. Arrows with solid lines represent “property” relation. Double-headed arrows represent “equals” relation.
- * According to these models, our system can generate and answer yes-no questions like4:

- (2) a. Question: Do the police believe that the thieves were trying to steal a solar panel from Sarah’s tin roof?
- Answer: Yes.
- (2) b. Question: In reality, were the thieves trying to steal a solar panel from Sarah’s tin roof?
- Answer: Not sure.
- (2) c. Question: Before little red-cap was surprised, did she think that the cottage door was open?
- Answer: No.
- (2) d. Question: Does little red-cap think the cottage door is open now?
- Answer: Yes.

• **Parser:**

– **Pre-processing:**

- * The pre-processing component consists of a sentence detector, a tokenizer, a chunker, and an anaphora annotator

– **Psych-term-argument Parsing:**

- * We first produce a set of mental state operations, one per psych-term-argument

- * SRL is used to annotate each sentence with PropBank argument labels
- * The set of psych-terms are chosen from a larger set of mental expressions drawn automatically from the WordNet along the synset (sets of synonyms) links.
 - The seed words used for collecting mental expressions from the WordNet contains 6 mental verbs: “think”, “want”, “pretend”, “confess”, “surprise” and “realize”.
 - For each mental expression returned by WordNet, we restrict the next search depth to 3. Using this method, WordNet returns 238 different verbs and phrases, among which we choose 42 psych-terms that are relatively less ambiguous for our initial system development.
- * Following parses are obtained using SRL for the example sentences 1(a) and 1(b) above.
 - (3) a. 0: [ARG0 The police] [TARGET believe] [ARG1 the thieves were trying to steal a solar panel from Sarah ’s tin roof]
 - (3) b. 0: The police believe [ARG0 the thieves] were [TARGET trying] [ARG1 to steal a solar panel from Sarah ’s tin roof]
- * The psych-terms are matched to the text annotated by SRL. A set of grammar rules are used to map the target verbs and their arguments to mental operations below:
 - (4) a. (new-single-modal {The police} ’((new-statement {the thieves were trying to steal a solar panel from Sarah’s tin roof})) {belief})
 - (4) b. (new-single-modal {the thieves} ’((new-statement {to steal a solar panel from Sarah’s tin roof})) {intention}))

– Building Nested Mental Operation:

- * Flat **mental state operations** are combined to form nested **mental state operations** based on how the corresponding source text is nested
 - (5) (new-single-modal {The police} ’((new-single-modal {the thieves} ’((new-statement {to steal a solar panel from Sarah’s tin roof})) {intention})) {belief}))
- * These nested mental state operations are used in building hierarchical representation of the mental states

• Mental State Representation as a Semantic Network

- We use **Scone** KB to store the mental states as **mental contexts** nodes.
- **Scone** can be viewed as a semantic network representation, with nodes representing entities and links representing relations or statements tied to these entities

• Why Mental Context?

- Mental context allow us to organize information efficiently

- * nested contexts and inheritance avoid the need of copying information as contexts evolve (due to events)

– mental interactions can be viewed as communication across contexts

• **Psych-Terms:**

- Psych-term is a subject-predicate pair in natural language (NL), e.g. realize, tell etc.
- We use SRL to parse NL to identify psych-terms and various other roles
- A Psych-term corresponds to an event from its agents point of view.

- * This event leads to the evolution of the agent's mental context, i.e. the event acts as a connector of 2 mental contexts, one before the event and one after the event (which evolves from the earlier one).

– **Predicates (actions) of a Psych-term:**

- * Each action/predicate in Scone has a type (aka description/frame/sense) with type-restricted slots (or roles) plus some statements about those roles and their relationships.

- e.g. In "John told X to Mary", "tell" has slots for a speaker (John in this case) and a listener (Mary) whose mental context gets updates with X, which could be one or a collection of statements (bundled together in the evolving mental context of Mary)

- This instance of "tell" has been cloned from the description (frame) of "tell" with appropriate slots fill in (i.e. with John, Mary and X)

- * Some psych-terms have complex semantics and need to be decomposed into atomic operations

- For example, one sense of the word “pretend” can be represented as “X is not true in reality and person P1's belief, but P1 wants person P2 to believe it”.

- This example involves several contexts: the reality, the belief of P1, the intention of P1, the belief of P2 under the intention of P1, as well as the before context and the after context of “pretend”.

- Notice that there can be other psychological terms (e.g. “want”) in the definition of “pretend”, which involves other mental context operations.

• **Scone:**

- A KB used for storing our model
- It is a semantic network of entities (as nodes) and relations (as links)

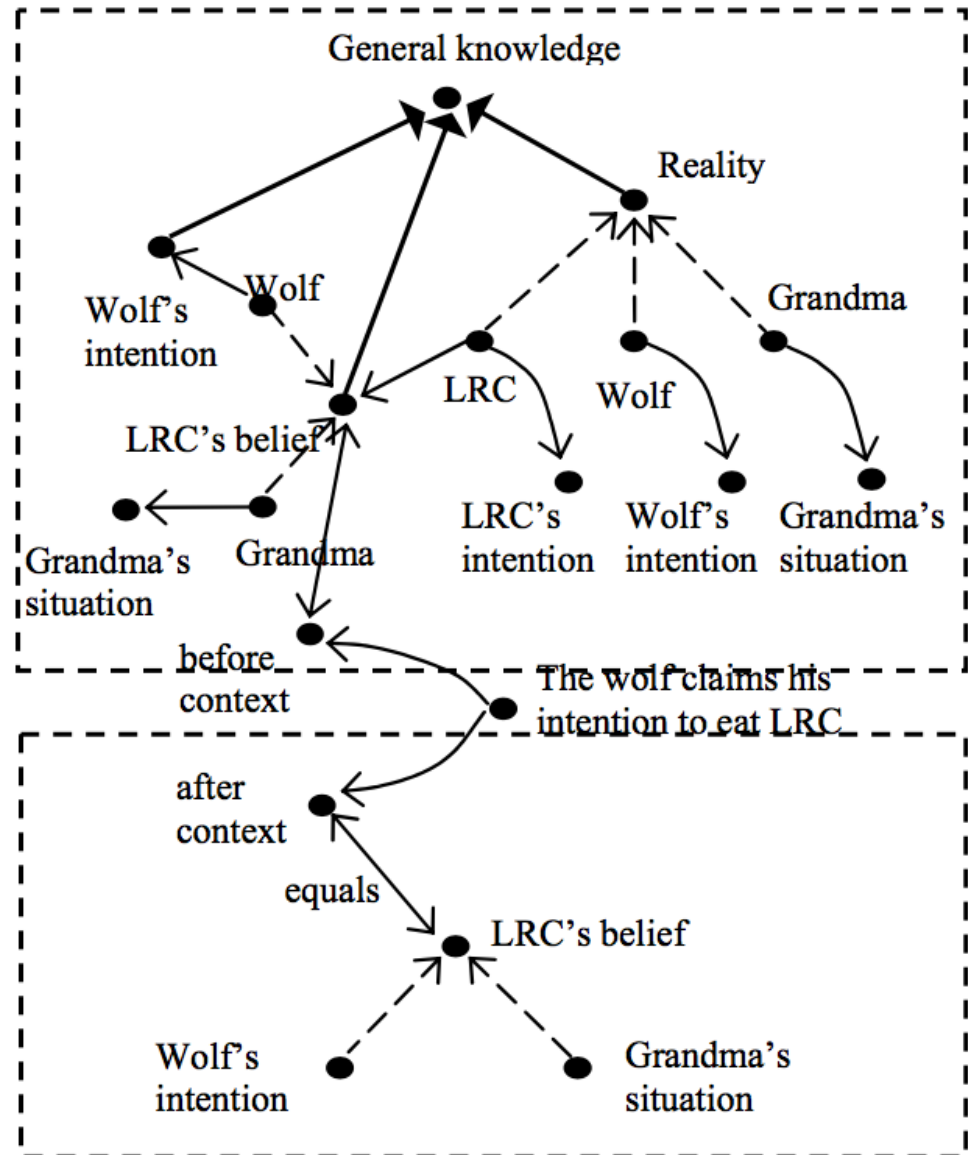
- * Entity is not necessarily an object but also could be a type or a specific instance of an action

– **Scoping:**

- * Each node (link) in the network is tied to a context node in which the node's entity (link's relation) is true (i.e. active)

- In Scone, mental contexts are modeled as roles attached to entities.
 - * For example, little red-cap's belief is a belief context role of little red-cap. Meanwhile, she can have different belief contexts at different times
- **Example:** The figure below shows the context network corresponding to the "little redcap story" below:

Little red-cap's mother told her to take some food to her grandmother and warned her not to run off the path. A wolf met little red-cap on her way. He suggested that she runs deep into the wood to pick flowers. When little red-cap arrived at her grandmother's home, she felt strange because her grandmother's appearance changed a lot. It turned out the wolf pretended to be her grandmother . . .



- * In this figure, Arrows with big solid heads represent “sub-context” relation. Double-headed arrows represent “equals” relation. Arrows with dotted lines represent “in-context” relation.
- * The figure shows that little redcap has multiple types of mental contexts:

- She has her belief about:
 - wolf's intention, which is a sub-context
 - grandma's situation, which is another sub-context
 - She has her perception of reality (which is inherited as a sub-context of the reality context)
 - Her belief in the after context of the event where "the wolf claims his intention to eat little red cap".
- * The figure also depicts the before and after context of the event where "the wolf claims his intention to eat little red cap".
 - * The figure also depicts wolf's real intent, grandma's real situation and LRC's own intent
- From the figure above, we learn 3 important things:
 - * By default, mental context inherit from general context or a particular context for the story.
 - * Mental contexts are organized by events. Each psych-term is mapped to one mental event. The mental event update the mental context of the agent (of the event).
 - * Mental contexts are environment sensitive, e.g. the wolf's intent is different from the little red caps belief of the wolf's intent.

- **Model Accuracy:**

- We can do inference on the Scone KB from the context nodes generated from text. We can generate yes-no questions from this inference. The correctness of the questions tell us how accurate our model is.
- Please see the paper for the evaluation of this model.

- **References:**

- Understanding Mental States in Natural Language - Chen - CMU - 2009 - c3
 - * Local Copy

(b) Modeling Mental Context and their interactions

- **Introduction:**

- The goal of this paper is to **understand a story** which requires modeling human cognition from the point of view of its **characters** (aka agents)
 - * This in turn requires modeling mental states (aka world-view, mental attitude) of all the characters of the story
 - **mental state:** belief, supposition, intent, perceived reality, advice, states of knowledge

- also situations expressed by tenses (past, present, future)
- Mental state of each character is modeled using a **mental context**
- We also need to model **interactions** b/w mental contexts
- **Goals:**
 - Modeling general concepts of mental attitudes/actions like regretting and realizing
 - Modeling reactive changes: how cognition grows with gradually available knowledge (and how agents react to it)
 - Representing mental states as descriptions within mental contexts
 - * Use of context activation scheme for reasoning within contexts
 - Model multi-agent reasoning scheme from purely single agent's point of view at any one time (which seems closer to human cognition)
 - * The paper's model captures an evolving world from each agent's point of view (e.g. person P2's intent in person P1's view)
- **Modeling Inter-Contextual Interactions**
 - **Types of interactions:**
 - **Mental states don't interact directly:**
 - * In our model, one person does not directly interact with another person's mental states. Instead, he builds a model of the other's mental states according to his perception of the reality
 - **Inter-Contextual Rules:**
 - * We constrain the behaviors of different mental contexts under different mental events using inter-contextual rules.
 - * Once a mental event happens, the related mental contexts would check and modify their own structures and contents based on the new information.
 - Usually this self-adjustment can be achieved by a realization of a difference between the external world and the belief, assumption or expectation. According to this, newly updated mental contexts would be constructed.
 - A simple rule: when a conflict is detected between the perceived reality and mental contexts, build new beliefs according to the perceived reality.
 - Figure below shows an example of how little redcap changes her mind about the world according to the wolf's intention and actions.

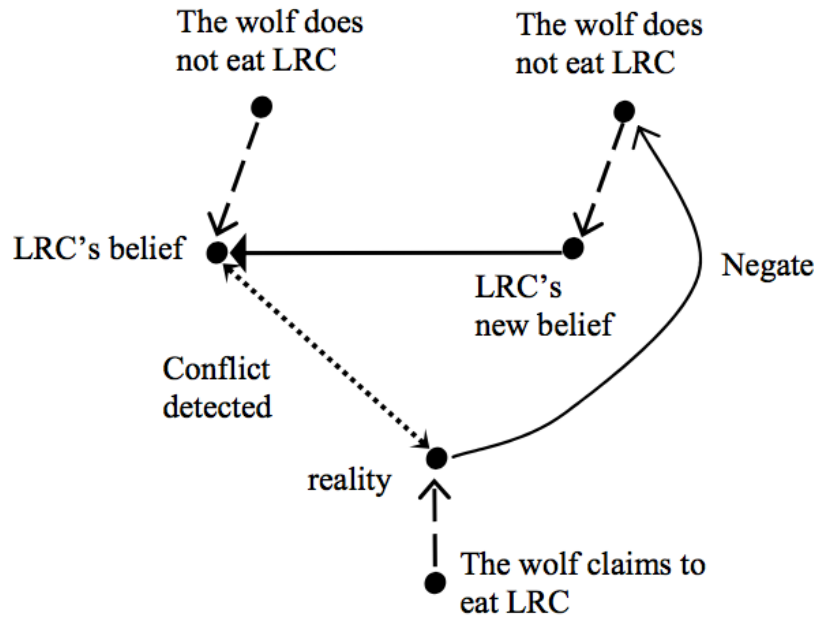


Figure 3. Inter-Contextual Activities.

- **Reference**

- Modeling Mental Contexts and Their Interactions - Fahlman et al - CMU - AAAI 2008

- * Local Copy

(c) Generating Instruction Automatically for the Reading Strategy of Self-Questioning

- **Goal:** To automatically generate Qs from narrative stories (e.g. red riding hood story is a narrative story)

- The idea is to teach kids self-questioning for effective comprehension

- The best place to insert questions while reading a narrative is after statements about character's mental state, such as belief, intention, supposition, emotion etc.

- Such statements typically draw inferential connections between key story elements, thereby providing opportunities to pose good **why** questions

- Mental states are identified by modal verbs as explained in the previous 2 papers

- About 239 modal verbs were identified

- **Question Generation**

- 3 types of Qs are generated:

- * What, Why & How

- Narrative is transport into a situation model encoded as a semantic network as explained in the 2 previous papers

- Qs templates and a morphology generator are used to transform inferred information in the situation model into Qs.

- * Morphology generator is for transforming the verb appropriately based on tense

- A question template consists of a question word and required information for that question type.

- * The question templates are:

- What did <character> <verb>?
 - Why/How did <character> <verb> <complement>?
 - Why was/were <character> <past-participle>?

- * Characters, verbs, participles, and complements come from the situation model.

- * Examples from the story below:

Little red-cap's mother told her to take some food to her grandmother and *warned* her not to run off the path. A wolf met little red-cap on her way. He *suggested* that she runs deep into the wood to pick flowers. When little red-cap arrived at her grandmother's home she was *surprised* to see the door open. She also *felt* strange on seeing her grandmother because the later's appearance had changed a lot. It turned out the wolf *pretended* to be her grandmother ...

- What did wolf suggest to little red-cap ?
 - How did LRC feel when she arrived at grandmother's home ?
 - Why was LRC surprised ?

- **References:**

- Generating Instruction Automatically for the Reading Strategy of Self-Questioning - Mostow et al - AIED 2009 - c69

- * Local Copy

(d) Generating Questions Automatically from Informational Text

- **Goal:** Generate Qs from informational text

- **Differences with Narrative Text:**

- Informational text has different text structure. Compare the 2 examples below, the first is from narrative text and the second is from informational text

- * (1) Peter thought it best to go away without speaking to the white cat.

- * (2) Rainbows are seen after it rains and the sun is out.

- * As exemplified by sentence (1), narrative text involves characters, their behavior, and mental states that drive it.

- * In contrast, informational text does not require characters.

- * Mental states do occur in informational text but with (slightly) less frequency than narrative text.
- * In addition, it places more emphasis on descriptions and explanations, which are often used to introduce objective phenomena, as in sentence (2).
 - Thus informational text is characterized by causal relationship (through conditions and temporal context) and modality (i.e. possibility and necessity)

– **Conditional and Temporal context:**

- * describe causation
- * Examples:
 - see (2) above
 - (3) If humans removed all the kelp from the sea soon all the other sea life would start to suffer as well.
- * Linguistic expressions which indicate conditional context
 - if, even if, only if, as long as etc.
- * Temporal expressions are identified by ARG-TMP label with SRL
 - 4 Types of temporal expressions are observed in the training data:
 - a general condition such as “after it rains and the sun is out,”
 - a date or time such as “in 1999,”
 - a duration of time such as “for several hours,”
 - and a rhetorical relationship (at the same time) such as “while she was reading.”
 - **Filtering**
 - Here we focus only on the first type of temporal expression, which tends to indicate causality. We filter out the other 3 types using regular expressions and other mechanisms

– **Modality: possibility and necessity:**

- * Expressed by Auxiliary verbs
- * Types of Auxiliary verbs:
 - Hypothetical: would
 - predictive: will, could
 - prescriptive: should, must, ought to, shall
- * Example
 - (4) All goats should have covered shelters where they can escape the weather

– **Building Mental Contexts (aka situation model)**

- * Uses 6 rules depending on the semantic categories (conditional, temporal or mental) of the text. The rules are used to build various sub-contexts and store elements of text in those sub-contexts
- * Sample Rule:
 - create a temporal context to store the when-statement; re-order existing temporal contexts based on time order.

– **Question Generation:**

- * Use 4 templates
- * For conditional context:
 - What would happen if <x>?
- * For temporal context, we used two templates:
 - When would <x>?
 - What happens <temporal-expression>?
- * For linguistic modality:
 - Why <auxiliary-verb> <x>?"
 - Here <x> maps to semantic roles tagged with ARG0 (the agent), TARGET (the verb), ARG1 (the theme), and ARG2, if any.
- * Since we aimed at questions about general conditions, which do not concern tense, we included auxiliary verbs such as “would” in the question templates. Therefore, we do not need morphology generation for verbs, as we did for narrative text questions.
- * Examples generated from the sentences (2), (3) & (4):
 - When would rainbows be seen?
 - What happens after it rains and the sun is out?
 - What would happen if humans removed all the kelp from the sea?
 - Why should all goats have covered shelters?

• **Results:** See the paper

• **References:**

- Generating Questions Automatically from Informational Text - Chen et al - 2009 - c1
- * Local Copy

3.2 RNN Approaches

3.2.1 QG From KB Triples

1. Generating Factoid Questions with RNN: The 30M Factoid Q-A Corpus

- **Reference:**

- Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus - Bengio's Group - ACL 2016 - c36

- * Local Copy

- **Introduction:** Uses MT inspired technique to do Q generation from Freebase triples

- **Task Definition:** Given a freebase triple (s, r, o), a question is phrased using the subject s and relation r and the answer being the object o.

- The question might contain hints relating to object o, the answer, e.g.

- * "What city is the American actress X from?" contains the hint that the answer (which is an object in the triple) is of category city and it is in America

- * These kind of hints are often prevalent in Qs generated by human annotaters, e.g. in the crowd sourced **SimpleQuestions** datasets we use for training

- **Datasets:**

- **SimpleQuestions** for training of the RNNs

- Embeddings for entities and relations are trained using **Freebase Triples** using TransE (a mechanism to compute relational embeddings)

- **Model:** A lossy translation from structured knowledge to Q in human language where the object part of the knowledge is intentionally left out.

- **Encoder:**

- * Maps the atoms (i.e. s, r & o) of a triple into a fixed size vector.

- It first maps each atomic to a vector using an Embedding table

- The embedding is learned separately (directly from Freebase Triples) using TransE (and hence the embedding matrix are not parameters of this model)

- Next a linear projection is taken for each vector derived in the previous step

- Finally all the 3 projections are concatenated to give a fixed size vector

- **Decoder:**

- * GRU is used for the decoder RNN.

- * At every time step of the decoder attention is used to weight the 3 parts of the vector coming from the encoding stage

- **Modelling the Source Language**

- * The embedding table used in the encoder stage for embedding atoms from the (s, r, o) triples are learnt by training a multi-relational embedding-based model, TransE, on Freebase

– Generating Qs

- * There is the issue of data sparsity and handling unseen words (which were not encountered in training) on the decoder side.
- * This is handled by heuristically identifying words in the Q (during training) which correspond to the subject and replacing them by the <placeholder> token.
 - The model is trained using modified Qs (with <placeholder> tokens instead of actual words from the subject atom)
 - The idea is to generate a <placeholder> token in the Q as a proxy for the subject. This token is replaced by the subject of the triple at the test time.
 - Example:
 - Given the fact {fires creek, contained by, nantahala national forest} the original question "Which forest is Fires Creek in?" is transformed into the question "Which forest is <placeholder> in?" at the training time.
 - Note that we do not use <placeholder> tokens in the input language (the encoder side) because then the entities and relationships in the input would not be able to transmit semantic (e.g. topical) information to the decoder
 - If we had included <placeholder> tokens in the input language, the model would not be able to generate informative words regarding the subject in the question (e.g. it would be impossible for the model to learn that the subject Paris may be accompanied by the words French city)
 - **Multi-placeholder (MP) Model:** a variant of the placeholder model, where we use 60 different placeholder tokens such that the placeholder for a given question is chosen based on the subject category extracted from the relationship.
- * This way of handling the output leads to a vocabulary of only 7000 words whose embeddings can be learnt as part of our model

• Baseline:

- For our experiments, we use a (non-parametric) template-based baseline model for comparison with the model presented in this paper.
- It uses Qs from the SimpleQuestions datasets and their corresponding Freebase facts as the training set.
- Each Q in the training set is modified s.t. the sequence of words corresponding to the subject is replaced by the <placeholder> token (same as in the section for **Generating Qs** above)
- During testing, given a fact F , we pick a fact F_c from the training set, uniformly at random from all the facts with the same relation as in F . We take the Q corresponding to F_c and modify its <placeholder> token with the subject from F

• Training Procedure

- Theano, Adam
- Dimensionality of TransE embeddings on the encoder side: 200
- Dimensionality of word embeddings on the decoder side: 200
- Learning rate for all NN models: 0.00025
- Gradient Clipping: 0.1
- Dimensionality of the hidden state of decoder RNN: 600
- **Evaluation:** Bleu, Meteor, sentence similarity metric and Human Evaluation.
- **Results:** On Blue, Meteor, and sentence similarity metrics, the RNN model proposed in this papers beats the baseline.
 - Human evaluation study:
 - * Here for each fact, a pair of Qs were shown to human, where the pairing is done b/w human-baseline, baseline-RNN or RNN-human. The humans are asked to determine which Q in each pair is more fluent.
 - * The RNN generated Qs were judged as the better candidate in more than 50% of the pairs in both the baseline-RNN as well as RNN-human cases.

3.2.2 QG from Document conditioned on answer

- Machine Comprehension by Text-to-Text Neural Question Generation - Microsoft Montreal - 2017
 - Local Copy

3.2.3 More

- Learning to Ask: Neural Question Generation for Reading Comprehension - Cardie et al - ACL 2017 - c3
- WikiReading: A Novel Large-scale Language Understanding Task over Wikipedia - Google Research - ACL 2016 - c12
- Creativity: Generating Diverse Questions using Variational Autoencoders - Schwing et al - UIUC - 2017
- Learning to Disambiguate by Asking Discriminative Questions - Li et al - CMU - 2017
- Crowdsourcing Multiple Choice Science Questions - Weibel et al - UCL/Allen Inst - 2017
- Insight on deciding difficulty of Q:
 - Knowledge Questions from Knowledge Graphs - Seyler et al - UIUC - 2016
- Question Answering and Question Generation as Dual Tasks - Microsoft Research Asia - 2017
- Neural Models for Key Phrase Detection and Question Generation - Montreal U/ Microsoft Maluba - 2017
- Neural Question Generation from Text: A Preliminary Study - Furu Wei et al - Microsoft Research - 2017
- Question Generation from a Knowledge Base with Web Exploration - Song et al - 2017

3.3 Other Approaches

- Deep Questions without Deep Understanding - Labutov et al - ACL 2015 - c11
 - Create a crowd-sourced set of Q templates relevant to a set of (wikipedia article categories, sections) combinations
 - * Example categories: people, location etc.
 - A large part of wikipedia article belong to top 8 categories
 - * Examples of section: childhood, honors, later life etc.
 - * Top 2 categories (people, location) and their combinations with 50 sections cover a very large part of wikipedia.
 - * Crowd Sourcing Methodology
 - Present content of an article/section and ask the turk to design upto 10 Q templates
 - Crowd-sourcing of relevance of generated Qs (using (category, sections) specific Q templates) on new article sections
 - Labeling an unlabeled text segment: Use the text in an article to generate (category, section) labelling if the segment is from an article whose (category, section) label is not known.
 - * Use LR on features extracted from the text of the text segment.
 - * Use labeled wikipedia articles as training/test data
 - Rank Q templates specific to (category, section) label for a text segment using a binary LR classifier trained on crowd-sourced relevance data
 - Use all templates with predicted probability above a certain threshold and use those to generate Qs at test time.
 - Gives pretty good precision and recall
- Question Generation from Concept Maps - Olney et al - D&D 2012 - c17
 - This generates Qs which span multiple sentences in the source text
- Semantics-based Question Generation and Implementation - Yao et al - 2012

4 Misc

- Learning to Compose Neural Networks for Question Answering - 2016
- Building a Large Annotated Corpus of English: The Penn Treebank - Marcus - 1993 - c5887

4.1 Definitions / Summarization

4.1.1 Important Papers

- Answering Definitional Questions: A Hybrid Approach - 2004
- Distributed Representations of Sentences and Documents - 2014 - 859c
- A Neural Attention Model for Abstractive Sentence Summarization - EMNLP 2015 - Facebook - c177
- Selective Encoding for Abstractive Sentence Summarization - Furu Wei - ACL 2017
- Learning-Based Single-Document Summarization with Compression and Anaphoricity Constraints - UC Berkeley - ACL 2016
 - include content
- A Deep Reinforced Model for Abstractive Summarization - Socher et. al - 2017
- Get To The Point: Summarization with Pointer-Generator Networks - Manning et al - 2017

4.1.2 DataSets

4.2 Visual QA

- VQA: Visual Question Answering - c226 - 2015 (read this first to understand data)
- Stacked Attention Networks for Image Question Answering - Yang - c94 - 2015 (read this next since this a very good yet simple model)
- Dynamic Memory Networks for Visual and Textual Question Answering - Socher et. al - ICML 2016 (read this 3rd as state-of-art)
- Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual QA - Goyal - 2016

4.3 Dialog Generation

4.3.1 Benchmark Dataset

Evaluating Prerequisite Qualities for End-to-End Dialog Systems

4.3.2 Learning End-to-End Goal-Oriented Dialog

- Learning End-to-End Goal-Oriented Dialog

4.3.3 Sequence-to-Sequence Models

1. Background Information

- seq2seq paper
- Neural Conversation Model

2. diversity of info

- A Diversity-Promoting Objective Function for Neural Conversation Models

3. persona & consistency of info

- A Persona-Based Neural Conversation Model

4. multi-context problem modeled as reinforcement learning

- Deep Reinforcement Learning for Dialogue Generation

5. Spoken Dialog System

- Partially observable Markov decision processes for spoken dialog systems

6. References

- slides

4.3.4 Spoken Dialog

- POMDP-based Statistical Spoken Dialogue Systems: a Review - Young - 2013
 - local copy