Core techniques of **QA Systems over KBs a Survey**

Core techniques of
QA Systems over KBs a Survey
Guillermo Echegoyen Blanco

Core techniques of **QA Systems over KBs** a **Survey**

- A Question Answering System should be able to: *Understand a Natural Language Question so as to be able to answer based on some pre-known data.*
- Typically involves accepting a question and generating a SparQL query capable of extracting the information which answers the user question.
- QALD benchmark
- WebQuestions benchmark
- SimpleQuestions benchmark

Perspective taken by most of the QA Systems evaluated on QALD Systems are tested like black boxes, so it is not clear enough with techniques work well on each part.

└Tasks

Tasks

- Question Analysis
- Phrase Mapping
- Disambiguation
- Query Construction

Question Analysis #1

Analyze syntactic features to extract meaningful information:

- Type of question
- Multilinguality
- Correspondance to KB entities/classes.
- Tokens in the sentence and it's relations.
- Useless words in the sentence.

- (is it a Which, What. . . question).
- (is it in English, French. . .).

Question Analysis #2

Techniques based on:
- Recognizing Named Entities
- Segmenting with *POS** Tags
- Identifying dependencies using parsers

POS Tag: Part-Of-Speech Tag

- Recognizing Named Entities consists in finding the entities corresponding to parts of the phrase (eg: Europe dbr:European_Union):Which token correspond to which resource in the KB
- Segmenting is like tokenization of different parts of the string, where the tag is usually universal
- Dependencies refer to parts of the phrase which depend upon others, direct cumpliment, adjective, subjective noun...

Question Analysis #3 - Recognizing named entities

Identify Named Entities and map to resource in KB
- *NER* Tools: Tools from NLP, **Standford NER Tool**. Domain specific, **low precision 51%** (**he2014a**)
- *N-Gram*: Map n-grams to KB entities. Adv: Each NE can be recognized in the KB, disadv: Dissambiguation explodes (**too much candidates**). (SINA: **shekarpour2015a**, CASIA: **he2014a**)
- *Entity Linking* Tools: **DBpedia Spotlight** (**daiber2013a**), **DBpedia Lookup** and **AIDA** (**yosef2011a**). Recognize NE and find the underlying KB resource, dissambiguating on the way. Adv: All-in-one. Disadv: Limited service, **KB dependant**.

Identify tokens in the sentence that refer to a resource in the KB, discarding useless words.

- When grouping n-grams, if an entity is found, the n-gram is considered, else more n-grams are tried.

Propose n-grams with attention mechanism?

Question Analysis #4 - Segmenting using POS Tagging

Identify which phrase correspond to instances, properties, classes... and which is irrelevant.

- *Handmade rules*: Regular expressions depenending on question type, structure.... (PowerAqua **lopez2012a**, Treo **freitas2014a**, DEANNA **yahya2013a**). Disadv: **regex built by hand**.

WRB	VBD	DT	NNP	NNP	VBN	.
When	was	the	European	Union	founded	?

Figure: POS tagging from the **Standford POS Tagger**

The general strategy with POS tags is to identify some reliable POS tags to recognize entities relations and classes. Regex over those POS tags

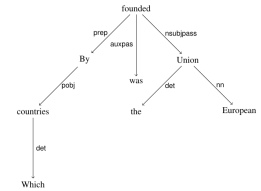
Combine POS tags, NER tags, no handmade rules

- *Learning rules*: **Machine Learning** approach, train over corpus (Xser **xu2014a**, UTQA **pouran2016a**, very good results). Disadv: **training corpus needed**.

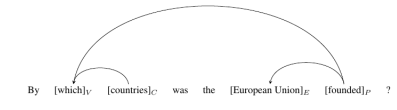
none	V-B	C-B	none	none	E-B	E-I	R-B	;
By	which	countries	was	the	European	Union	founded	?

Figure: Question annonated with **CoNLL IOB format**

Dependency grammars: **Standford dependency parser**, word dependencies. Adv: can extract relations along with it's arguments (gAnswer **zou2014a**, **PATTY nakashole2012a**)



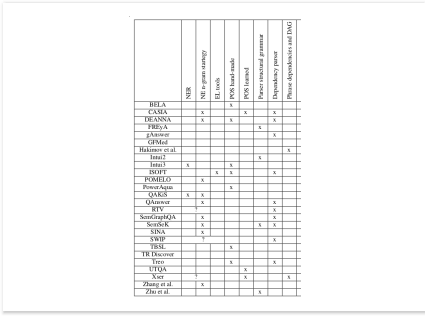
Phrase Dependencies and DAGs: Dependencies between phrases. SHIFT-REDUCED parser. Disadv: **parser trained on dataset** (Xser **xu2014a**).



- DAG based parser operates on a phrase level, dependency grammars on a word level.
- DAG uses POS tags as features

Core techniques of **QA Systems over KBs** a **Survey**

Question Analysis #8 - Summary



	NER	NE = guess starting?	EL tools	POS based tools	POS learned	Parser (structural) grammar	Dependency parser	Phrase dependencies and DAG
BELA			x					
CASIA		x		x		x		
DEANNA		x		x		x		
FREyA					x			
gAnswer						x		
GFMed								
Hakimov et al.								x
Intui2					x			
Intui3	x			x				
ISOFT			x	x		x		
PARALEX								
Pomelo Aqua		x		x				
QAKiS	x	x						
QAnswer		x				x		
RTV		?				x		
SemGraphQA		x				x		
SenseK		x				x	x	
SINA		x						
SWIP		?				x		
TBSL		?		x				
TR Discover								
Treo				x			x	
UTQA					x			
Xser		?			x			x
Zhang et al.		x						
Zhu et al.						x		

Which techniques to choose?

- Xser (**trained DAG**) reports best results on *QALD 4.1 & 5*
- gAnswer (**Dependency grammars**) reports fastest results on *QALD 3 & 4*
- UTQA (**Learned POS tags**) reports best results on *QALD 6*

Machine Learning approach: Can be fast enough and there is plenty of data available.

Phrase Mapping #1

Find the resources in the KB with the highest probability that maps to the phrase.

Problems:
- String similarity
- Semantic similarity
- Language

- In this phase: consider the phrase mapping problem when s and label(r) have only a semantic relation
- String similarity: very similar words, different meaning (which, witch)
- Semantic similarity: words with related semantic meaning but different writing (king, queen)
- PATTY is used in Xser

Phrase Mapping #2

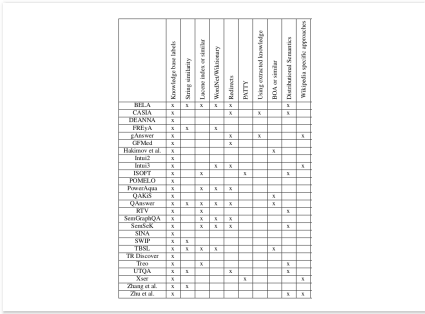
- Database with lexicalization: *WordNet*, *Wiktionary*, *PATTY* Expand the phrase with synonims and use that for search. Adv: High number of candidates, disadv: **Big search space**, **not very useful for domain specific mappings**.
- Mappings using large texts: **word2vec** semantics reflected in the associated vector. Adv: aids in **lexical gap, string similarity and semantic similarity**, disadv: **needs training on large texts, noisy, performance**.

- PATTY is a database with relational lexicalization, uses pattern synsets (is album, [[num]] album by)
- A possible advantage of using PATTY is that response text could be "easily" constructed.

Phrase Mapping #3

- KB Labels: Search in the labels provided by the KB's entitiy (all)
- Redirects: Follow the *owl:sameAs* links (gAnswer **zou2014a**)
- Extracted knowledge: From the previous phase (gAnswer **zou2014a**). Relations and arguments.
- Wikipedia specific: **DBPedia Lookup**, **Wikimedia Miner Tool** (gAnswer **zou2014a**, Xser **xu2014a**, **zhu-a**)

Labels can be a powerful resource, smart indexing techniques can be used here (gAnswer)



	Knowledge/Free labels	String similarity	Lucene index or similar	WordNet/Wiktionary	Rulelists	PATTY	Using extracted knowledge	BOA or similar	Distributional Semantics	Wikipedia specific approaches
BELA	x	x	x	x	x				x	
CASIA	x				x		x		x	
DEANNA	x									
FREyA	x	x		x						
gAnswer	x				x		x			x
GFMed	x				x					
Hakimov et al.	x							x		
Intui2	x									
Intui3	x			x	x					x
ISOFT	x		x			x			x	
POMELO	x									
PowerAqua	x		x	x	x					
QAKiS	x							x		
Qanswer	x	x	x	x	x			x		
RTV	x		x						x	
SemGraphQA	x		x	x	x					
SenseSeK	x		x	x	x				x	
SINA	x									
SWIP	x	x								
TBSL	x	x	x	x				x		
TR Discover	x									
Treo	x		x						x	
UTQA	x	x			x				x	
Xser	x					x				x
Zhang et al.	x	x								
Zhu et al.	x								x	x

- Distributional semantics word2vec
- Indexed labels: Lucene index

Which techniques to choose?

The best results here depend on the previous step and the computing resources available.
Options (can be combined together):

- KB's Labels
- Redirects
- word2vec
- PATTY

Mixed approach with all the possible methods?? Maximize:

- performance
- KB independance

Disambiguation #1

QA systems generate lots of possible interpretations due to language ambiguities and search process.
- Find univocally the resource that maps to the requested question.

Base approach (local dissambiguation):
- String or semantic similarity to resource label.
- Consistency check between the properties and their arguments.

- String or semantic similarity to resource: (include)
- Consistency check between the properties and their arguments: (exclude).
- Local dissambiguation excluded, all systems do it. Example "Who is the director of The Lord Of the Rings?", with no information associated with the director resource, it is not possible.

Disambiguation #2 - Graph Search

Dissambiguation carried out in the KB search step

- Subgraph matching against the KB (**gAnswer zou2014a** does it on phrase mapping). Represent the question as a dependency graph and find an isomorfic subgraph in KB. Adv: very fast. Disadv: dissambiguation carries over. (**high precision, low recall**)
- **SemSek aggarwal2012a** and **Treo freitas2014a** do it only with recognized instances (during question analysis phase). (**low precision, high recall**)

Assume that all relational phrases can be deduced from the question.

- gAnswer scores each possible match proportionally to the distance between labels and resources, searches both in edges and nodes.
- PowerAqua explores only the most probable mappings, doing a balance between recall and precision based on the question analysis.
- SemSek and Treo do not explore the relations, but the attached properties of the different interpretations on the KB found matches.

gAnswer searches in the edges and vertices, **SemSek, Treo** search on instances and properties attached.

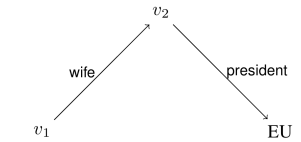
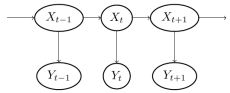


Figure: Subgraph generated for the question "Who is the wife of the president of the EU?"

Dissambiguation #4 - Hidden Markov Model (HMM)

"By which countries was the EU founded?"
Two stochastic processes:

- **Hidden (dissambiguation)** $X_{t\in N}$: {*dbo:Country, dbr:Euro, dbr:European_Union, dbp:founded, dbp:establishedEvent*}.
- **Observed (question tokens),** $Y_{t\in N}$: {*"countries", "EU", "founded"*}.



- Assumption: This means that the appearance of a resource at time t depends only on the appearance of a resource at t -1
- Random variables

Dissambiguation #5 - HMM

The problem is reduced to find the most probable set of states. Extra parameters:

- Initial probability $P(X_0 = x)$ for $x \in X$
- Transition probability $P(X_t = x_1 | X_{t-1} = x_2)$ for $x_1, x_2 \in X$
- Emission probability $P(Y_t = y | X_t = x)$ for $x \in X, y \in Y$

It is not necessary to know the the dependency between different resources, just the available resources.

Disambiguation #6 - HMM

SINA (**shekarpour2015a**): **slow**

- Emission: string similarity between label and segment.
- Initial & Transition: estimated based on the distance of the resource in the KB and popularity.

RTV (**giannone2013a**): **inaccurate**

- Emission: word embeddings
- Initial & Transition: uniform across all resources

The slow part is the distance to the resource in KB (first approach), biiiig search space

Dissambiguation #7 - ILP & MLN

ILP Optimization problem

- **DEANNA** (**yahya2013a**) Dependencies between the segments have to be computed in the question analysis phase. **slow, low precision & recall**

Markov Logic Network

- **CASIA** (**he2014a**) Hard constraints like ILP, soft constraints flexibility **training needed low precision & recall**

Ambiguity during phrase mapping and segmentation
ILP: Dependencies between the segments have to be computed in the question analysis phase.
Boolean variables to indicate:

- if a segment of a question is chosen or not
- if a resource corresponding to a segment is chosen
- if a segment corresponds to a property or an instance

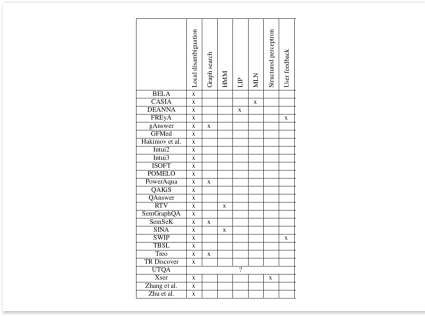
Optimization function terms:

- Increase if the label of a rsource is similar to the corresponding segment
- Increase if the two selected resources often occur in the same context
- Maximize the number of selected segments.

Considering:

- Similarity of the phrase and the corresponding resource
- Popularity of a label for a resource
- Compatibility of the range and domain of a property with the arguments.

Xser (**xu2014a**) Solves ambiguity **fast, training needed**



	Local disambiguation	Graph search	HMM	LIP	MLN	Structured perceptron	User feedback
BELA	x						
CASIA	x				x		
DEANNA	x			x			
FREyA	x						x
gAnswer	x	x					
GFMed	x						
Hakimov et al.	x						
Intui2	x						
Intui3	x						
ISOFT	x						
POMELO	x						
PowerAqua	x	x					
QAKiS	x						
QAnswer	x						
RTV	x		x				
SemGraphQA	x						
SenseKA	x	x					
SINA	x		x				
SWIP	x						x
TBSL	x						
Treo	x	x					
TR Discover	x						
UTQA	?						
Xser	x					x	
Zhang et al.	x						
Zhu et al.	x						

Which techniques to choose

- Best results *QALD 6*, reported from UTQA (**pouran2016a**), whose method is unkown.
- Fastest method is gAnser (**zou2014a**) which dissambiguates in the phrase mapping step. (**subgraph matchin**)
- Best results in *QALD 4.1 & 5* by Xser (**zou2014a**) which uses a (**Perceptron**)

Query Construction #1 - Issues

Construct a **SPARQL** query that reflects user question and gets the answer.
Semantic Gap: Issues with how the information is encoded in the KB. One cannot deduce how the information is stored from the question.

"Which countries are in the European Union?"
Could be encoded as:
dbr:Greece dbp:member dbr:European_Union
dbr:France dbp:member dbr:European_Union
or as:
dbr:Greece dct:subj dbc:Member_states_of_the_European_Union
dbr:France dct:subj dbc:Member_states_of_the_European_Union
How to search correctly

Approaches:

- Using templates
- Using information from the question analysis
- Using Semantic Parsers
- Using Machine Learning
- Using semantic information

Templates with parts of the query to be filled, in general by triples.

- **QAKiS** (**cabrio2012a**) select queries with only one triple.
- **ISOFT** (**park2014a**) ASK over one triple, simple SELECT, COUNT and ORDER BY or FILTER.
- **PowerAqua** (**lopez2012a**) reduces the question to one or two triples ($<= 2$ predicates).

Very restricted questions, language is too rich, disambiguity is key

Query Construction #4 - Question Analysis

Most systems get the form of the query in the question analysis and phrase mapping step.

- **Freya, Intui3** (**dima2014a**) resources extracted in the phrase mapping step are combined into triples.
- **DEANNA** (**yahya2013a**) regex over POS tags in analysis step mapped to resources in phrase mapping step. ILP in dissambiguation step to get the triples.
- **gAnswer, QAnswer, RTV, SemGraphQA** (**zou2014a**, **ruseti2015a**, **giannone2013a**, **beaumont2015a**) extract all the possible information from the dependency graph.

Question analysis because is done in that step

- gAnswer: The graph takes the form of the final query, resources associated with nodes and edges are fetched from the KB and used in the query.
- QAnswer: Scan dependency tree to find subgraph tokens corresponding to resouces, many graphs, local dissambiguation to get the best ones. Top ranked graph chosen for query.
- RTV dependency graph -¿ ordered list of alternated properties and non properties. Resources searched and disambiguated with HMM.

Special mention to Xser (best results), next slide

Xser (**xu2014a**) 3 ML algorithms, two KB independant (on the question analysis phase), one KB dependant (on dissambiguation step)

- First algorithm: determines segments of the question corresponding to variables, properties, instances and classes.
- Second algorithm: find dependencies between phrases. (**Standford dependencies, PATTY**)
- Third algorithm: Dissambiguation with a **Structured Perceptron**

The problem with these methods is that they all **assume** that **is possible to deduce the structure of the SPARQL query from the structure of the question** without knowing how the knowledge is encoded in the KB.

Compose a grammar and use it to extract structure from the query.

- **G**rammatical **F**ramework (**GFMed marginean2017a**)
- **F**eature-based **C**ontext **F**ree **G**rammar (**TR Discover**, **song2015a**)
- **C**ombinatorial **C**ategorial **G**rammar (**hakimov2015a**)
- **L**exical **T**ree **A**djoint **G**rammar (**TBSL unger2012a**, **BELA walter2012a**)

Question has to be well formulated. For each lexical item a corresponding semantic representation is needed. (ie married has to map with *dbo:spouse*). Learning corpus (**hakimov2015a**) or from POS tags (**unger2012a**). In general, **low recall**

Lexical item	Syntactic category	Semantic representation
Barack Obama	NP	$\texttt{dbr:Barack\_Obama}$
is	$(S\backslash NP)/(S\backslash NP)$	$\lambda f.\lambda x.\texttt{f(x)}$
married to	$(S\backslash NP)/NP$	$\lambda y.\lambda x.\texttt{dbo:spouse(x,y)}$
Michelle Obama	NP	$\texttt{dbr:Michelle\_Obama}$

Figure: Semantic parsed question "*Barack Obama is married to Michelle Obama*"

CASIA (**he2014a**) (**low recall & precision**)

- Question Analysis step: extract features like position of a phrase and POS tags or the type of dependency in the dependency tree
- Phrase Mapping step: associate resources with phrase segments and extract more features
- Dissambiguation step: MLN with extracted features to find most probable relation between segments and most probable mapping. **retrained for each KB**

SINA (**shekarpour2015a**), **POMELO** (**hamon2014a**), **zhang2016a** do not rely on the syntactic features of the question, instead the whole process is done based on the KB, just with semantic information.
Advantages:

- **high recall, & precision**

Disadvantages:

- **computationally expensive**
- **does not respect user question syntax**. No difference between *"Who is the mother of Angela Merkel?"* and *"Angela Merkel is the mother of who?"*

Core techniques of **QA Systems over KBs a Survey**

Query Construction

Query Construction #11 - Summary

Query Construction #11 - Summary

	Using templates	Using info. from the QA	Using Semantic Parsing	Using machine learning	Semantic information	Not generating SPARQL
BELA			x			
CASIA				x		
DEANNA		x				
PREsA		x				
gAnswer		x				
GFMed			x			
Hakimov et al.			x			
Intui2		x				
Intui3		x				
ISOFT	x					
POMELO					x	
PowerAqua	x					
QALD	x					
QAnswer		x				
RTV		x				
ScoreGraphQA		x				
SemSeK						x
SINA					x	
SWIP	x					
TBSL			x			
Treo						x
TR Discover			x			
UTQA				?		
Xser		x				
Zhang et al.					x	
Zhu et al.						x

Which techniques to choose?

There is no clear way to do this. A good approach is to construct the query on the previous analysis steps, assuming structure can be extracted from the question.

Conclusions

There exists many techniques for each part:

- Smart balance between different techniques lead to best results
- Tendency goes to Machine Learning

When possible, maximize:

- KB independance (pluggability)
- Performance (real time, scalability)
- Extracted Knowledge (implicit, enriches the context/domain)

- Data availability grows every day
- Domain and context are key, big future for attention based mechanisms