

Leveraging Community-built Knowledge For Type Coercion In Question Answering

Aditya Kalyanpur, J William Murdock, James Fan and Chris Welty

Mehdi Allahyari

CSCS8380

Spring 2012



Introduction

- ▶ Typing: recognizing whether a given entity is a member of a given class
 - ▶ is a fundamental problem in AI areas
- ▶ Focus on the typing problem where both the entity and type are expressed lexically (as strings)
- ▶ Traditional approach: QA systems relied on *Predictive Annotation*:
 - ▶ a fixed set of expected answer types are identified through manual analysis of a domain

INTRODUCTION

- ▶ a background corpus is automatically annotated with possible mentions of these types before answering questions.
- ▶ system then analyzes incoming questions for the expected answer type, mapping it into the fixed set used to annotate the corpus, and restrict candidate answers retrieved from the corpus to those that match this answer type using semantic search.

Disadvantages

- ▶ restricts the answer types to a fixed and typically small set of concepts.
 - ▶ not work well when answer types expressed using a variety of lexical expressions.
- ▶ the QA system performance is highly dependent on the precision and recall of the predictive annotation software used.
- ▶ Approach called type-and-generate.

Generate And Type Approach

- ▶ candidate answers are initially produced without use of answer type information.
- ▶ subsequent stages check whether the candidate answer's type can be *coerced* into the Lexical Answer Type (LAT) of the question.
 - ▶ LAT: terms in the question that indicate what type of entity is being asked for.
- ▶ The framework is based on the notion of Type Coercion (TyCor).

Advantages

- ▶ does not rely on a fixed type system, however it does not discard one when available and useful.
- ▶ it is a multistrategy and multi-source approach, gathering and evaluating evidence in a generative way rather than a predictive one.
- ▶ analyzes and scores candidate answers and produces for each candidate answer a probability that it is (or is not) of the right type.

Background - Open Domain QA

- ▶ Watson is a QA system capable of answering open-domain questions on the challenging TV quiz show Jeopardy!
- ▶ questions cover a wide range of topics and are expressed using rich, complex natural language expressions.
- ▶ nearly any word in the English language can be used as an answer type in Jeopardy! questions

Watson Component - DeepQA

- ▶ massively parallel probabilistic evidence-based architecture designed to answer open domain natural language questions. Consists of 4 stages:
 - I. **Question Analysis:** performs a detailed analysis to identify key characteristics of the question (such as focus, lexical answer type, etc).
 - ▶ Focus is the part of the question that refers to the answer, and typically encompasses the string representing the lexical answer type (LAT).

I. Question Analysis

- ▶ system employs:
 - ▶ various lexico-syntactic rules for focus and LAT detection.
 - ▶ also uses a statistical machine-learning model to refine the LAT(s).
- ▶ LAT detection includes a confidence, and all type scores are combined with LAT confidence.


2. Hypothesis (Candidate) Generation

- ▶ system issues queries derived from question analysis to search its background information (data- and knowledge-bases) for relevant content.
- ▶ uses a variety of candidate generators to produce a list of potential answers.

3. Hypothesis And Evidence Scoring

- ▶ all candidates, regardless of how they were generated, are evaluated.
- ▶ different algorithms and sources are used to collect and score evidence for each candidate answer.
- ▶ Type information is just one kind of evidence that is used for scoring. Others: temporal/spatial constraints, etc.

4. Candidate Ranking

- ▶ machine-learning models are used to weigh the analyzed evidence and rank the answer candidates and produce a confidence.
 - ▶ models generate a confidence that each answer candidate is the correct answer to the given question, and the system answers with the top-ranked candidate.
 - ▶ system can also choose to refrain from answering if it has a low confidence in all of its candidates.
- 

Type Coercion (TyCor) Framework

- ▶ is part of Hypothesis and Evidence scoring, and consists of answer scoring components that each take a Lexical Answer Type (LAT) and a candidate answer, and return a probability that the candidate's type is the LAT.
- ▶ TyCor performs four steps:
 - ▶ Entity Disambiguation and Matching (EDM)
 - ▶ Predicate Disambiguation and Matching (PDM)
 - ▶ Type Retrieval (TR)
 - ▶ Type Alignment

Entity Disambiguation And Matching (EDM)

- ▶ uses an existing source of typing information to find the entity in that source that corresponds to the candidate answer.
- ▶ the candidate is just a string so this step must account for both polysemy (the same name may refer to many entities) and synonymy (the same entity may have multiple names).
- ▶ Each source may require its own special EDM implementations. For example DBpedia encodes useful naming information in the entity URI.

Predicate Disambiguation And Matching (PDM)

- ▶ Similar to EDM, looking for the type in the source that corresponds to the LAT.
- ▶ Usually the same algorithm as EDM.
- ▶ If using unstructured information as a source, the PDM step just returns the LAT itself.

Type Retrieval (TR)

- ▶ After EDM, the types of the retrieved entity must be themselves be retrieved.
- ▶ This step may be different depends on if TyCore using structured or unstructured sources and may need some NLP.

Type Alignment

- ▶ The results of the PDM and TR steps must then be compared to determine the degree of match.
- ▶ In sources containing e.g. a type taxonomy, this includes checking the taxonomy for subsumption, disjointness, etc. For other sources, alignments utilize resources like WordNet for finding synonyms, hypernyms, etc. between the types.
- ▶ Each step generates a score reflecting the accuracy of its operation, The final score is a combination of the four step scores.

Community-built Knowledge For TyCor

- ▶ community-built knowledge resources could be effectively used to deal with the very long tail of answer types.
- ▶ So
 - ▶ structured knowledge base (DBpedia) and ontology (YAGO).
 - ▶ semi-structured folksonomies with wide topical coverage (Wikipedia Categories and Lists) is used.

DBpedia And YAGO

- ▶ A one-to-one correspondence between all Wikipedia pages and DBpedia entries.
- ▶ Additionally, DBpedia has type assertions assigned from a collection of ontologies, including YAGO.
- ▶ YAGO ontology has mappings to WordNet.
- ▶ design points of DBpedia and YAGO enable us to obtain precise type information for many instances.

TyCor Algorithms

- ▶ All the three TyCors use one algorithm:
 - ▶ **EDM Algorithm:** takes as input the candidate answer string and a corresponding context, and returns a ranked list of Wikipedia page URIs that match the candidate, with associated match scores.
 - ▶ five heuristics to compute the match scores:
 - I. Direct Contextual Match:
 - if the Wikipedia URI of the candidate is known, and EDM is not performed and use it as the result of our EDM step with a score of 1.0.

TyCor Algorithms

2. Title Match:

- ▶ When there is an exact string match between the candidate string and the title of a Wikipedia page, the URI is returned with a score of 1.0

3. Redirect Match:

- ▶ When the candidate string matches the name of a redirect page, the redirect destination URI is returned with a score of 1.0

TyCor Algorithms

4. Disambiguation Match:

- ▶ When the candidate string matches the title of a Wikipedia disambiguation page all the disambiguation URIs are returned with a score of $1/(\text{the number of disambiguations})$

5. Anchor-Link Match:

- ▶ When a candidate string matches one or more anchor text strings in Wikipedia, all the URIs pointed to by those anchors are returned with a score for each based on the conditional probability of the link pointer given the anchor text.

TyCor Algorithms

- 6. DBpedia name properties:
 - ▶ DBpedia includes over 100 name properties, properties whose objects are some form of name string (firstName, lastName, etc). When a candidate string matches one of these, the triple subject is returned with a score of $1/(\text{number of URIs returned})$.
 - ▶ EDM algorithm also contains an optional parameter to rank the results based on the *popularity* of the corresponding Wikipedia page, overriding the confidence set by the heuristics.

YAGO TyCor

- ▶ uses the EDM step described before, and transforms the Wikipedia page URLs returned at the end of the step to corresponding DBpedia URIs.
- ▶ Type Retrieval using Dbpedia:TR algorithm produces a set of URIs for the Yago types of the candidate entity. DBpedia contains type information for entities represented by the `rdf:type` relation that comes usually from YAGO.

YAGO TyCor

- ▶ PDM in YAGO:
 - ▶ PDM algorithm: produces a set of URIs for the Yago types that match the LAT, by matching the LAT to the labels or IDs of Yago types. We then score the matches based on a weighted combination of its WordNet sense rank, and the number of instances of the concept in DBpedia.
- ▶ YAGO Type Alignment: produces a single score based on the alignment of the instance types from the TR step, and the LAT types from the PDM step.

YAGO Type Alignment conditions

- ▶ Equivalent/Subclass match: 1.0
- ▶ Disjoint match: -1.0
- ▶ Sibling match: 0.5
- ▶ Superclass match: When the instance type is a superclass (hypernym) of the LAT type a score of 0.3 is returned.
- ▶ Statistical Relatedness: 0.25
- ▶ Lowest Common Ancestor (LCA): When the LCA of the instance type and LAT type is deep in the taxonomy, score of 0.25 is returned.

Wiki-category And Wiki-list Tycors

- ▶ Both TyCors use the same EDM component as YAGO TyCor.
- ▶ They also both use a simple lookup in an RDF store for Type Retrieval that returns the category names for entity.
- ▶ both have a trivial Predicate Disambiguation and Matching step that simply returns the LAT itself.
- ▶ Type Alignment: terms are matched using a variety of resources such as WordNet.

Experiments-Evaluating EDM On Wikipedia Link Anchors

- ▶ data set is comprised of 20,000 random pairs of Wikipedia anchor texts and their destination links.
- ▶ The destination of an anchor text is a Wikipedia article whose title (string) may or may not explicitly match the anchor text.
- ▶ tested four versions of EDM, with and without popularity ranking and DBpedia name properties.

Evaluating EDM On Wikipedia Link Anchors

Ranking	Names	Precision	Recall	Avg. # of candidates
No	No	74.6%	94.3%	16.97
No	Yes	74.7%	94.3%	13.02
Yes	Yes	75.2%	94.3%	16.97
Yes	No	75.7%	94.3%	13.02

Table 4. EDM performance on 20,000 Wikipedia anchor texts

- ▶ DBpedia names increase the number of candidates without impacting precision or recall.
- ▶ This is partially a side-effect of the link-anchor based evaluation, which skews the results to prefer alternate names that have been used as link anchor texts.

Evaluating Typing On Ground Truth

- ▶ TyCor components' impact on end-to-end QA performance can be measured through tests, but they do not reflect how well the components do at the task of type checking.
 - ▶ Because wrong answers may also be instances of the LAT.
- ▶ So To measure the TyCor components performance on the task of entity-typing alone:
 - ▶ manually annotated the top 10 candidate answer strings produced by Watson from 1,615 Jeopardy! questions, to see whether the candidate answer is of the same type as the lexical answer type.

Evaluating Typing On Ground Truth

- ▶ the resulting data set contains a total of 25,991 instances, because some questions contain multiple LATs.
 - ▶ 17,384 (67%) of these are negative

Tycor Component	Accuracy	Precision	Recall
Yago Tycor	76.9%	64.5%	67.0%
Wikipedia Category Tycor	76.1%	64.1%	62.9%
List Tycor	73.3%	71.9%	31.6%
All Three (Union)	73.5%	58.4%	69.5%

- ▶ Shows the performance of the three TyCor components that use community–built knowledge, by counting any candidate with a TyCor score > 0.0 to be a positive judgment.

Impact On End-to-end Question Answering

	No TyCor	YAGO TyCor	Wiki- Category TyCor	Wiki- List TyCor	All 3 TyCors
Baseline Accuracy	50.1%	54.4% (+4.3%)	54.7% (+4.6%)	53.8% (+3.7%)	56.5% (+6.5%)
Watson Accuracy	65.6%	68.6% (+3.0%)	67.1% (+1.5%)	67.4% (+1.8%)	69.0% (+3.4%)

- accuracy of the DeepQA question answering system with different TyCor configurations.

Conclusion

- ▶ novel open-domain type coercion framework for QA that overcomes issues associated with Predictive Annotation techniques.
- ▶ TyCor framework four key steps: EDM, PDM, TR and TA.
- ▶ community-built knowledge resources can be effectively integrated into this TyCor framework and provide corresponding algorithms for the four TyCor steps.

Conclusion

- ▶ results show that the TyCors built using Web knowledge resources perform well on the EDM and entity typing tasks
 - ▶ both fundamental issues in NLP and Knowledge Acquisition.
- ▶ significantly improves the end-to-end QA performance of the Watson system on rich and complex natural language questions taken from Jeopardy!

THANKS!