Interactive Text Graph Mining with a Prolog-based Dialog Engine

Paul Tarau and Eduardo Blanco

January 20, 2020

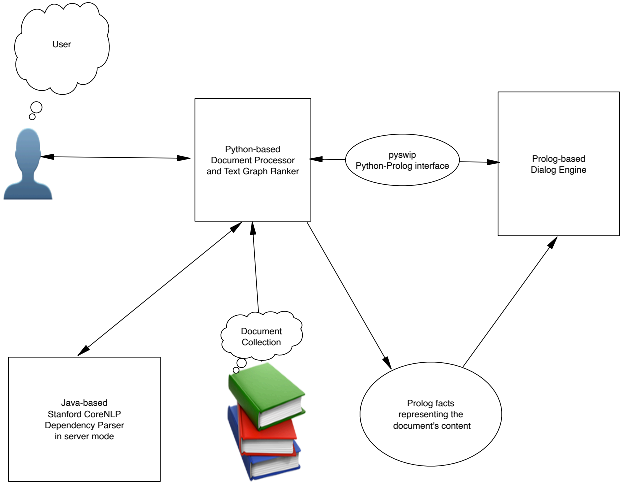
University of North Texas

PADL'2020

Overview

- we start from a deep-learning based dependency parser
- digest with it a book or a scientific document
- derive a text graph using dependency links
- we runPageRank on the graph and then generate a Prolog database
- together, these provide summary, keyphrase and relation extraction
- we connect it our Prolog-based dialog agent
- ⇒ interactive exploration of a document's most salient content elements
- ⇒ synergy between deep-learning and symbolic AI: each doing what they are best at

Overview of the system architecture



User
Python-based Document Processor and Text Graph Ranker
pyswip Python-Prolog interface
Prolog-based Dialog Engine
Document Collection
Java-based Stanford CoreNLP Dependency Parser in server mode
Prolog facts representing the document's content

Figure: Key system components and their interactions

Ranking the text graphs

- inspired by the effectiveness of algorithms like Google's PageRank, recursive ranking algorithms applied to text graphs have enabled extraction of keyphrases, summaries and relations
- they shed a holistic view on the interconnections between text units that act as recommenders for the most relevant ones
- originally (TextRank), text graphs were produced from colocations in a sliding window for words, or overlapping bag words for sentences
- today's neurally-trained dependency parsers produce highly accurate dependency trees (and also, POS-tags, lemmas, NERs)
- $\Rightarrow$ 'distilled" building blocks through which a graph-based natural language processing system
- $\Rightarrow$ our nodes: sentences and their words/lemmas, connected with edges derived from dependency links

Deriving the dependency graph of a sentence

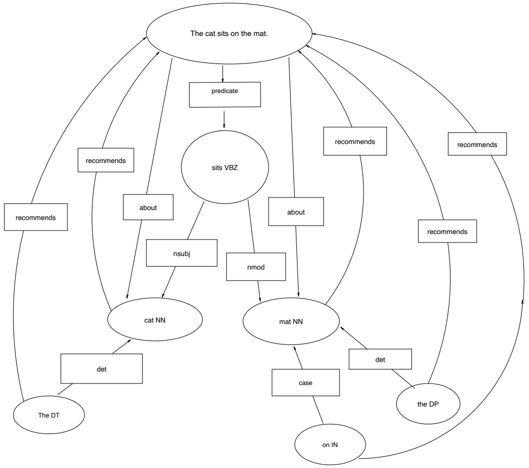


Figure: Dependency graph with redirected and newly added arrows

Some (elective) graph surgery

- we design a unified algorithm to obtain graph representations of documents, that are suitable for keyphrase extraction, summarization and interactive content exploration
- we use unique sentence identifiers and unique lemmas as nodes
- we reverse some links in the dependency graph provided by the parser, to prioritize nouns and deprioritize verbs
- we redirect the dependency edges toward nouns with subject and object roles (*"about"* edges)
- we create *"recommend"* links from words to the sentence identifiers and back from sentences to verbs with *predicate* roles
- we ensure that sentences recommend and are recommended by their content
- $\Rightarrow$ we use PageRank (or *personalized PageRank*, when answering a query) to rank the most important words and sentences

Graph of a document: just the tip of the iceberg

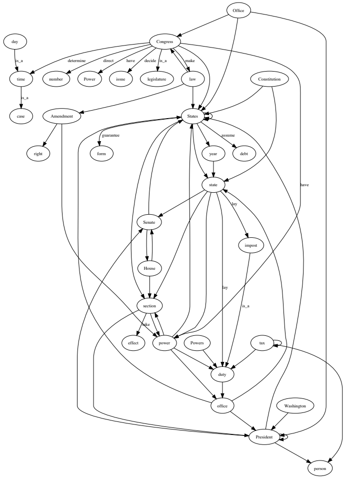


Figure: Text graph of the highest ranked words in the U.S. Constitution

Summary and keyphrase extraction

- as link configurations tend to favor very long sentences, a post-ranking normalization is applied for sentence ranking
- after ordering sentences by rank we extract the highest ranked ones and reorder them in their natural order in the text to form a more coherent summary
- we use the parser's compound phrase tags to fuse along dependency links
- our keyphrase synthesis algorithm ensures that highly ranked words will pull out their contexts from sentences, to make up meaningful keyphrases
- we mine for a context of 2-4 dependency linked words of a highly ranked noun, while ensuring that the context itself has a high-enough rank
- we compute a weighted average favoring the noun over the elements of its context

Relation extraction

- subject-verb-object relations extracted from the highest ranked dependency links
- "is-a" and "part-of" relations using WordNet (hypernyms/hyponyms, meronyms/holonyms)
- by constraining the two ends of an "is-a" or "part-of" edge to occur in the document, we avoid entities unrelated to the document's content
- additional relations, if a domain-specific ontology is available

The generated Prolog database

1. `keyword(WordPhrase).` – the extracted keyphrases
2. `summary(SentenceId,SentWords).` – summary sentences with identifiers and words
3. `dep(SentID,From,FromTag,Label,To,ToTag).` – dependency link, with the first argument indicating the sentence
4. `edge(SentID,From,FromTag,Relation,To,ToTag).` – edge marked with sentence identifiers indicating where it was extracted from, and the lemmas with their POS tags at the two ends of the edge
5. `rank(LemmaOrSentenceId,Rank).` – the computed rank
6. `w2l(Word,Lemma,Tag).` – a map associating to each word a lemma, as found by the POS tagger
7. `svo(Subject,Verb,Object,SentenceId).` – subject-verb-object relations extracted from parser input labels in verb position or WordNet-based `is_a` and `part_of` relations
8. `sent(SentId,SentWords).` – sentence identifier and the list of words

The scientific document dataset

- the resulting logic program can be processed in Prolog, possibly enhanced by using constraint solvers and abductive reasoners or via SAT and ASP systems
- possible benefits from such extensions for tackling computationally difficult problems like word-sense disambiguation (WSD) or entailment inference
- domain-specific reasoning (e.g., with action languages, epistemic extensions etc.)
- the *Krapivin document set*: a collection of **2304** research papers annotated with the authors' own keyphrases and abstracts
- the resulting 3.5 GB *Prolog dataset* at `http://www.cse.unt.edu/~tarau/datasets/PrologDeepRankDataset.zip` is made available to researchers interested to explore declarative reasoning or text mining mechanisms

The dialog engine

- we use *pyswip* from Python to initiate the SWI-Prolog process, load the database and manage the dialog loop
- the dialog agent associated to a document answers queries as sets of salient sentences extracted from the text, via a specialization of our summarization algorithm to the context inferred from the query
- the dialog starts by displaying the summary and keyphrases extracted from the document *and also speak them out if the* `quiet` *flag is off.*
- as part of an dialog loop, we generate for each query sentence, a set of predicates that are passed to the Prolog process, from where answers will come back via the `pyswip` interface
- the predicates extracted from a query have the same structure as the database representing the content of the complete document

The answer generation algorithm: query expansion

- query-expansion mechanism via relations that are derived by finding, for lemmas in the query, WordNet hypernyms, hyponyms, meronyms and holonyms, as well as by directly extracting them from the query's dependency links
- we use ranking from both the main document and the query
- we prioritize the highest ranked sentences connected to the highest ranked nodes in the query

The short-term dialog memory

- we keep representations of recent queries in memory, as well as the answers generated for them
- if the representation of the current query overlaps with a past one, we use content in the past query's database to extend query expansion to cover edges originating from that query
- overlapping is detected via shared edges between noun or verb nodes between the query graphs

The answer generation algorithm: sentence selection

- *personalized PageRank*, with a dictionary provided by highest ranking lemmas and their ranks in the query's graph, followed by reranking the document's graph to specialize to the query's content
- matching guided by SVO-relations
- matching of edges in the query graph against edges in the document graph
- query expansion guided by rankings in both the query graph and the document graph
- matching guided by a selection of related content components in the short-term dialog memory window
- when NER is available, matching directed by **where, when, who, etc.** questions trigger scanning the named entity database, also sent to Prolog

Querying the U.S. Constitution

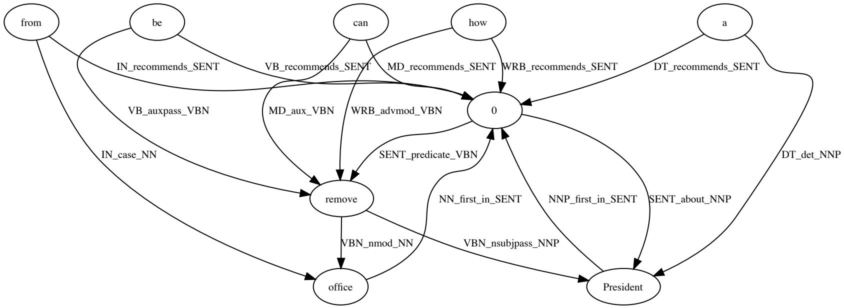


Figure: Graph of a query on the U.S. Constitution

Interacting with the dialog engine: the answers

»> *talk_about('examples/const')*

?− **How can a President be removed from office?**

59 : In Case of the Removal of the President from Office , or of his Death , Resignation , or Inability to discharge the Powers and Duties of the said Office , the same shall devolve on the Vice President ... .

66 : Section 4 The President , Vice President and all civil Officers of the United States , shall be removed from Office on Impeachment for , and Conviction of , Treason , Bribery , or other high Crimes and Misdemeanors .

190 : If the Congress , within twenty one days after receipt of the latter written declaration , or , if Congress is not in session , within twenty one days after Congress is required to assemble , determines by two thirds vote of both Houses that the President is unable to discharge the powers and duties of his office , the Vice President shall continue to discharge the same as Acting President ; otherwise , the President shall resume the powers and duties of his office .

Specializing w.r.t. a query: Personalized PageRank

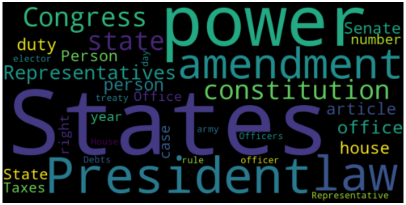
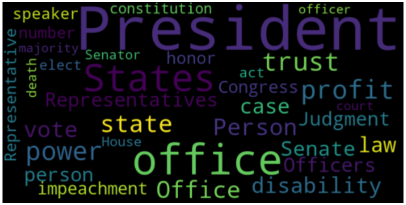


Figure: Word-clouds of the document before and after



Talking to Einstein's 1920 book on relativity

»> *talk_about('examples/relativity')*

?− **What happens to light in the presence of gravitational fields?**

611 : In the example of the transmission of light just dealt with , we have seen that the general theory of relativity enables us to derive theoretically the influence of a gravitational field on the course of natural processes , the laws of which are already known when a gravitational field is absent .

764 : On the contrary , we arrived at the result that according to this latter theory the velocity of light must always depend on the co-ordinates when a gravitational field is present .

765 : In connection with a specific illustration in Section XXIII , we found that the presence of a gravitational field invalidates the definition of the coordinates and the time , which led us to our objective in the special theory of relativity .

What happens to light in the presence of gravitational fields?

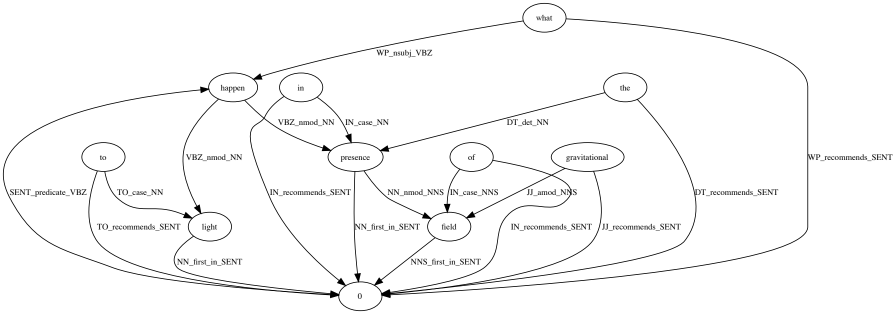


Figure: Graph of a query on Einstein's book on Relativity

Talking to Tesla (the car)

the User Manual is digested in 60 seconds, the queries are real time

```
»> talk_about('examples/tesla')
?- How may I have a flat tire repaired?
3207 : Arrange to have Model 3 transported to a Tesla Service Center ,
or to a nearby tire repair center .
3291 : Note : If a tire has been replaced or repaired using a different
tire sealant than the one available from Tesla , and a low tire pressure
is detected , it is possible that the tire sensor has been damaged .
```

- the highly relevant first answer is genuinely useful in this case, given that Tesla Model 3's do not have a spare tire
- being able to use voice queries while driving and in need of urgent technical information about one's car, hints towards obvious practical applications of our dialog engine

Conclusions

- work evolved from our search for synergies between symbolic AI and emerging machine-learning based natural language processing tools
- our text graph is based on heterogeneous, but syntactically and semantically meaningful text units (words and sentences), mutually recommending each other's highly ranked instances
- our Python-relation fact extraction, in combination with the Prolog interface, has elevated the syntactic information provided by dependency graphs with semantic elements ready to benefit from logic-based inference mechanisms
- given the standardization brought by the use of *Universal Dependencies*, our techniques are likely to be portable to a large number of languages
- the Prolog dataset at `http://www.cse.unt.edu/~tarau/datasets/PrologDeepRankDataset.zip`, derived from more than 2000 research papers, is made available to other researchers using logic programming based reasoners and content mining tools