1 Introduction

Richly annotated documents are a part of many knowledge extraction efforts. Such efforts may include knowledge base population (KBP), a task where it is important to recognize certain relationships between entities in text in order to add those facts to an entity-centric database. Document annotations might also serve as grist for training various distributional models of meaning.

In either setting, it is common for a research group to generate bulk annotations over a preferred corpus internally, using their own tools, programming languages and formats, but then reporting on this as merely an engineering pre-processing step not worth describing in significant detail. Worse, these annotated collections are often not available to the rest of the community, making it difficult to perform apples-to-apples comparison of the “real research”.

We previously sought to address these issues with the creation and release of the Annotated Gigaword corpus [1], a resource comprising 4.5 billion tokens of English newswire text, processed with a collection of then state-of-the-art tools from the community. The LDC distributed this collection [2] along with a Java utility library to easily work with the data in a variety of popular formats.

Here we describe a new effort that follows this same idea, with various extensions and improvements. Under the heading Concretely Annotated, we are processing a variety of standard corpora with multiple popular NLP tool-chains, collected together under a single data schema we have created that we refer to as CONCRETE. We envision a multimodal workflow, where, e.g., knowledge can be extracted from both text and audio. We developed CONCRETE to record and share annotations on structured human language data – both text and speech. Backed by Apache Thrift, this schema allows for direct programmatic access to the annotations across a variety of popular languages.

In this new data set, we include multiple tool outputs producing the same types of annotations; these annotations are represented together under a shared tokenization as different annotation theories. For example, these new resources include ACE ontology relations [3] – produced automatically by both BBN’s SERIF system [4] and an in-house JHU toolchain competitive to state-of-the-art (in prep) – as well as semantic frame analyses – produced by SEAMFOR [5] and an in-house semantic frame parser (in prep). As we demonstrate in this paper, CONCRETE’s common storage and annotation format allow multiple, independently-developed systems to be pipelined together. We also believe that developers will be able to leverage complementary aspects of repeated annotation types.

2 Facilitating Knowledge Extraction from Text

Many, including Chuch & Hanks (1990) [6] and DIRT [7], have considered the connections between shallower linguistic analyses and deeper semantic understanding ([8] [9] [10] [11] [12]). Multiple recent efforts, e.g., script learning [13], relation discovery [14], and information extraction [15], have utilized richer linguistic annotations on large corpora to improve knowledge extraction. As

\[\text{For example, all syntactic parse trees can be printed to a console as a phrase structure parse, whereas all NER labels are output in a CoNLL style.}\]
recent efforts in knowledge base population have shown [16], huge, highly annotated corpora can be leveraged to improve inference.

Multiple previous efforts have attempted to ensure the community has easy access to large and sufficiently annotated data. Annotated Gigaword [1] ran state-of-the-art tools on a single, large dataset of ten million documents in order to create a standard reference. ILLINOIS CLOUD NLP [17] developed a framework in which users could annotate individual large datasets on a remote, distributed compute grid, allowing annotated data to be created “on demand.” Goldberg and Orwant (2013) [18] released counts of “syntactic n-grams” – dependency paths spanning n words – from three million books, to facilitate PMI-inspired efforts. While these approaches, among others, all have their merits, our focus is on creating larger and richer common corpora that promote collaboration and have the potential to facilitate multimodal research.

In addition to adding semantic annotations, this work corrects some of the losses that arose from the Annotated Gigaword pipeline (AG). In AG there was a 2% loss in sentences arising to a bug in Splitta, which AG used to split sentences [19]. Second, coreference resolution only included non-singleton entities (those that have at least two mentions). Borrowing terminology from GAF [20], we make a distinction between (conceptual) instances of a situation or an entity, and the individual mentions of those instances in text. In NLP terms, this means we reify a coreference chain to stand for an entity, with the items in that coreference set the representative entity mentions. Therefore, we need to record singletons, even though suppressing singleton mentions is consistent with coreference evaluation [21]. To quantify the impact of these missing mentions, we employed a simple heuristic that matches NER spans with existing entities 2. We found that more than 1.6 million documents (16% of AG) would have had ten or more additional PER, LOC or ORG entities. This does not consider non-location or organization entities, which can be of interest in open-domain IE efforts.

3 Concrete

Concrete is an extensible data schema aimed at capturing a wide variety of natural language annotations over both structured and unstructured data. We sought a common format that reduces the complexity associated with both using different corpora (unique file formats or markup) as well as interfacing with or creating different annotation tools and analytics (programming languages and APIs). Some data types can be required to be present, providing some guarantees about what will be present in any given annotation layer and helping to prevent the accidental omission of critical information. These guarantees extend to the types of individual annotations, limiting a potential source of bugs. For instance Concrete annotation creators cannot accidentally populate a field with the wrong type of object, while Concrete readers do not need to manually cast values.

We further sought a representation amenable to multimodal data. While most of our current efforts have been focused on text processing, Concrete also supports audio data. We envision a multimodal workflow wherein automatic knowledge extraction is not limited to text corpora. The simultaneous interest that multiple fields share for grounded language tasks suggests that researchers will be looking for easy-to-use multimodal corpora and/or data exchange formats.

Implementation The Concrete schema is a set of Apache Thrift [22] schema files specifying 51 interworking structs. Structs contain instances of other structs, “primitive” types (e.g., strings and ints), and lists or maps of structs or primitives. Thrift generates schema-specific classes for many popular and widely used programming languages, including Java, Python, C++ and Javascript. The array of supported languages allows data to be shared easily across projects, lessening barriers toward collaborative efforts. Thrift also provides backwards compatibility: additions to the schema do not invalidate previously generated data, making previously annotated data compatible with future changes. Concrete was initially based on Google’s Protocol Buffer technology; we switched to Thrift to better align with related community standards such as the StreamCorpus format employed in NIST’s TREC KBA, or the ADEPT schema used in the ongoing DARPA DEFT program 4.

---

2 We compare a document’s maximal NER spans to every entity mention officially recorded in [1]. We report as “missing mentions” only those NER spans that do not overlap with a recorded mention.

3 In object-oriented languages, a struct becomes a class.

**Document Representation** CONCRETE defines a structure-preserving wrapper (called a Communication) around tokenized data. The tokenized data are realized by either a list or lattice of Tokens. In many text applications, the collection will be a TokenList, a flat sequence of Tokens; when capturing output from speech systems or machine translation, the collection may be better represented as a TokenLattice. All annotations are defined on these Tokens.

A Communication segments the observed data into different Sections, each of which maintains a list of Sentences. Every Sentence contains the collection of Tokens – either a TokenList or TokenLattice, encapsulated as a union record in a containing Tokenization struct. As explained shortly, Tokenizations allow us to succinctly keep related annotations together. As in \cite{23}, CONCRETE uses *global byte offsets* to index into the data: when representing text, we use character offset spans, and when representing audio, we use start and end time spans. Sections, Sentences and Tokens maintain these spans.

**Annotation Representations** CONCRETE allows both intrasentential and intersentential annotation theories. The former includes standard tagging tasks (e.g., part-of-speech, named entity recognition, lemmatization, cached out into TokenTaggings) and syntactic analyses (constituency or dependency parse). Because we store intrasentential annotations within every Tokenization, we keep, e.g., individual token labels close to the tokens themselves. By manipulating a single Tokenization, a developer has easy access to its different token taggings. We are not restricted in the types of TokenTaggings we can add: a tagging is agnostic to its actual meaning. We can just as easily store language ID for code-switched tweets as we can per-token sentiment.

Intersentential annotations revolve around mention identification and coreference, in a manner akin to \cite{20}, with the latter recording equivalent groupings in the former. CONCRETE implements mentions and coreference sets for both entities and situations – a term derived from the event semantics literature \cite{8, 24, 12} that broadly covers events, relations, facts, sentiments, and beliefs. Mentions point into a specific tokenization, recording both byte offsets and token indices. Although individuals are constrained to a specific Tokenization, coreference spans multiple Tokenizations. As a result, we store all mention and coreference theories as members of a Communication.

Crucially, a Communication may have the same type of annotation multiple times. For instance, each Tokenization may have multiple, competing NER or dependency parse theories, and a Communication may have any number of parallel or competing, e.g., entity coref theories.

**Open-Source Release** Our primary development languages are Java, Python and C++: as such, we have written utility libraries in these three languages. These libraries are in active development; they, along with the CONCRETE schema definition files, will be released as open source projects.\footnote{https://hltcoe.github.io/}

4 Corpora

We process four different corpora, covering formal newswire, casual internet media and Wikipedia. We mapped each corpus to CONCRETE, maintaining paragraph or section structure when possible.

**English Gigaword v5** The Gigaword corpus \cite{25} contains 4.5 billion words from 10 million English newswire articles, from 1994 - 2010. The data are sectioned into paragraphs. We maintain this structure and record the minimal metadata, such as dateline, within our CONCRETE schema.

**Annotated NYT** The New York Times Annotated Corpus \cite{26} is a collection of 1.8 million New York Times articles from 1987 - 2007. These articles have been heavily enriched with nearly fifty types of metadata by NYT staff. These metadata include a hierarchical taxonomic classification into different types of documents (e.g., Opinion vs. News/U.S./Rockies), an originating newsdesk (e.g., Business desk vs. Classifieds desk) and a list of prominent people, locations and organizations mentioned in the article.

**ColdStart** The 2014 ColdStart KBP track of NIST’s Text Analysis Conference (TAC) is based on a corpus of roughly 50,000 documents, with ~70% derived from the Gigaword corpus and the remainder a collection of more informal blog and discussion forum posts \cite{27}. This collection was preprocessed by the ColdStart organizers with BBN’s Serif tool and then released in order that participants would have equal access to the output of a competitive information extraction toolchain.
Wikipedia Previous work has shown that Wikipedia is a useful resource for knowledge base construction \cite{28, 29, 30}. We process Wikipedia in order to make available a text that is both encyclopedic and open access, akin to WaCkypedia\textsubscript{EN}, which followed from ukWaC \cite{31}.

5 Annotations

We process all corpora (§4) with four different suites: Stanford’s CoreNLP \cite{23}, BBN’s SERIF \cite{4}, CMU’s Semafor \cite{5} and an in-house JHU pipeline. We modified or developed each of these tools to read in and operate on the mapped CONCRETE data from above. See Table 1 for an overview of the different types of annotations each tool adds; note that a tool may add more multiple theories for each type of annotation, while multiple tools may add the same type of annotation.

In order for all annotations to exist in parallel, we force all tools to operate on a single (PTB-style) tokenization and sentence segmentation, obtained from the CoreNLP pipeline\cite{23}. We label individual tokens with two part-of-speech theories and three named entity theories (CoreNLP, SERIF and JHU \cite{32}), and a lemmatization theory (CoreNLP). We further use CoreNLP and SERIF to add constituency parses. These CoreNLP parses are converted to chunks using the same method as the CoNLL-2000 shared task \cite{33}. In total, three different tools produce five dependency parses: CoreNLP applies human-written rules to deterministically convert the Stanford constituency parse into the three types of Stanford dependencies \cite{34}, while CMU’s Stacked MST Parser \cite{35} (a variant of \cite{36}’s minimum spanning tree parser) and a fast, first-order dependency parser from JHU \cite{37} each provide a statistically determined dependency parse.

We record three types of situation mentions: frame analyses based on FrameNet \cite{9, 38, 39}, Propbank-style semantic role labeling \cite{40} and ACE relation extraction \cite{3}. We include frame analyses from both Semafor \cite{5, 41} and an in-house log-linear staged FrameNet parser (article in prep), where we tune the latter for high recall annotation. Semantic role labeling is done via a competitive system \cite{37, JHU}. We extract and record dual ACE relations using both SERIF and an in-house project that leverages word embeddings to be competitive with state-of-the-art (article in prep.). As a result of how “situations” are defined in the linguistics literature and in CONCRETE, most arguments of SituationMentions point to an EntityMention. All suites contribute to the parallel entity mention identification theories, but only SERIF and CoreNLP provide entity coreference atop their own mention identifications (singleton mentions included). The JHU entity mentions are ACE-style named entities from an in-house variant of prior work \cite{32}.

6 Utilities and Open-Source Release

We are releasing CONCRETE and utility libraries, including interfaces for Java, Python and C++, as open-source projects at \url{https://hltcoe.github.io/}. The Java and Python utility libraries can be easily downloaded or installed with Maven and PyPI, respectively. The above URL also has the data and annotations that can be freely-released (e.g., Wikipedia).

The language-specific interfaces and utilities are in active development, though they currently provide only basic annotation retrieval, e.g., providing Tokenizations for an EntityMention in

---

\footnote{6} We allow a destructive tokenizer; CoreNLP “Americanizes” words, e.g., replacing an “-our” suffix with “-or” in “colour”, for example. Certain non-content word normalizations help to alleviate token sparsity.
Figure 1: Example output from a command-line utility providing easy access to plain-text versions of CONCRETE annotations. 1a demonstrates Token annotations, plus a dependency parse, while 1b shows a constituency parse.

(a) CoNLL-style output.

```bash
$ ./concrete_inspect.py example.concrete
--pos --ner --lemmas --dependency
INDEX TOKEN LEMMA POS NER HEAD
----- ----- ----- --- --- ----
1 John John NNP PERSON 4
2 's 's POS O 1
3 daughter daughter NN O 4
4 Mary Mary NNP PERSON 5
5 expressed express VBD O 0
6 sorrow sorrow NN O 5
7 . . . O
```

(b) Constituency parse output.

```bash
$ ./concrete_inspect.py example.concrete
--treebank
(ROOT
(S (NP (NP (NNP John) (POS 's)) (NNP Mary)) (VP (VBD expressed) (NP (NN sorrow))))
)
```

which it appears. Due to their open-source nature, they can be extended to handle more complex tasks, such as intersecting multiple annotations.

Since the Thrift format is not viewable as plain-text, we have developed tools that allow researchers to quickly inspect a Communication at the command line. This utility also provides researchers with easy access to plain-text versions of CONCRETE annotations. See Figure 1 for an example. For instance, Token-level annotations can be output in a CoNLL-like format (1a), while constituency parses can be returned as standard S-expressions (1b). With this utility, existing systems can use the annotations provided in this paper without modifying code.

7 Conclusion

Our final data set contains more than approximately 400 million sentences. While at time of print the full annotation processing is in progress, approximately 1 TB of (non-compressed) annotations have been produced from initial corpora sizes of approximately 40 GB of uncompressed text. Fully processing the corpora through CORENLP took approximately 900 CPU-hours, or three days with 300 cores (≈115 tokens/second). The other tools we use range from 150 to 1,000 tokens/second.

Efforts such as DeepDive [16] exemplify the community’s interest in large scale knowledge base creation fostered by equally large text corpora processed with state-of-the-art NLP tools. Further, efforts such as [13, 42] in event and distributional semantics require a similar sort of processed input. As an effort to democratize such large scale efforts, we provide several different theories of token taggings, entity mentions, entities, and situations; researchers can investigate what particular tools are optimal for their use case, or more easily conduct systems combination research. Researchers can access the data by using the libraries mentioned earlier, or creating their own Thrift bindings for the programming language of their choice.

Acknowledgements Thank you to the three reviewers for helpful feedback, and to Johns Hopkins HLTCOE for providing support. We also thank Ed Loper, Mark Dredze and the participants of SCALE 2013 for feedback. Any opinions expressed in this work are those of the authors.

References


2Thank you to reviewer three for this example.


