Task-Dependent Visualization of Coreference Resolution Results

René Witte and Ting Tang
Fakultät für Informatik
Institut für Programmstrukturen und Datenorganisation (IPD)
Universität Karlsruhe (TH), Germany
witte@ipd.uka.de

Abstract

Graphical visualizations of coreference chains support a system developer in analyzing the behavior of a resolution algorithm. In this paper, we state explicit use cases for coreference chain visualizations and show how they can be resolved by transforming chains into other, standardized data formats, namely *Topic Maps* and *Ontologies*.

1 Introduction

The computation of coreference chains is an important task in natural language processing. Many high-level text analysis functions rely on coreferences, which makes it important to analyze the results of a particular resolution algorithm. The well-known coreference metrics like MUC [7] or CEAF [4] compute precision and recall values using a gold standard, which allows for a *quantitative* analysis of a system. However, these values only provide a conflated view of the performance; they do not allow for an in-depth analysis of the behavior of an algorithm, e.g., in order to find problematic entities that a coreferencer always "gets wrong." Especially when developing a rule-based or hybrid coreference resolution system, a *qualitative* analysis becomes important, focusing on individual chains and their entities in order to identify error sources.

Yet the sheer amount of data produced by a coreferencer even on a moderately-sized text makes it infeasible to rely on a tabular or matrix-like representation for understanding an algorithm's behavior. As humans are much better at analyzing images than numbers,[1] our idea is to transform coreference resolution results into dynamic graphical representations that can be explored and navigated by a user. Furthermore, coreference visualization should adapt to specific tasks, like chain and document navigation, error detection and analysis, or automatic summarization, in order to adequately support a developer.

However, graphical (2D/3D)-visualizations are notoriously difficult and costly to develop. Instead of building our own rendering pipelines from low-level graphical libraries, we investigated a different approach: The translation of coreference resolution results into standardized data formats, for which a multitude of visualization interfaces exist. This not only allows us to reuse existing graphical tools for NLP, but even permits the application of newly developed visualizations as long as they can read one of the standardized data formats we provide.

[1] See, e.g., [8]: *"Combining a computer-based information system with flexible human cognitive capabilities, such as pattern finding, and using a visualization as the interface between the two is far more powerful than an unaided human cognitive process."*

2 Use Cases for Visualization

Our premise is that a single, generic visualization cannot provide adequate support for the different, varying tasks concerning coreference chains in NLP. Consequently, our approach is to define specific *use cases* based on the work of an NLP system developer, which result in different, task-specific visualizations:

Chain and Document Navigation. The visualization should provide for both a quick overview of all created coreference chains (inter- and intra-document), as well as navigational aids to analyze the chain members. Cross-document chain visualizations should additionally provide cues for the document range they span.

Error Detection and Analysis. Analyzing the behavior of a coreference resolution algorithm is a major task during system development. A visualization that contrasts computed chains with a manual gold standard should allow a developer to identify "weak spots" in the algorithm's performance.

Automatic Summarization. Automatic summarization is an important application area of coreference chains and clusters. A visualization that shows summaries and their sentences together with the underlying coreferences can help the developer of a summarizer to discern and analyze the connections between a summary and its underlying coreferences.

3 Visualization Formats

As mentioned above, our goal is to transform coreference chains into external data formats that are supported by existing visualization tools. In this section, we first examine previous approaches to coreference visualization, and then discuss standard data formats for which suitable graphical tools exist.

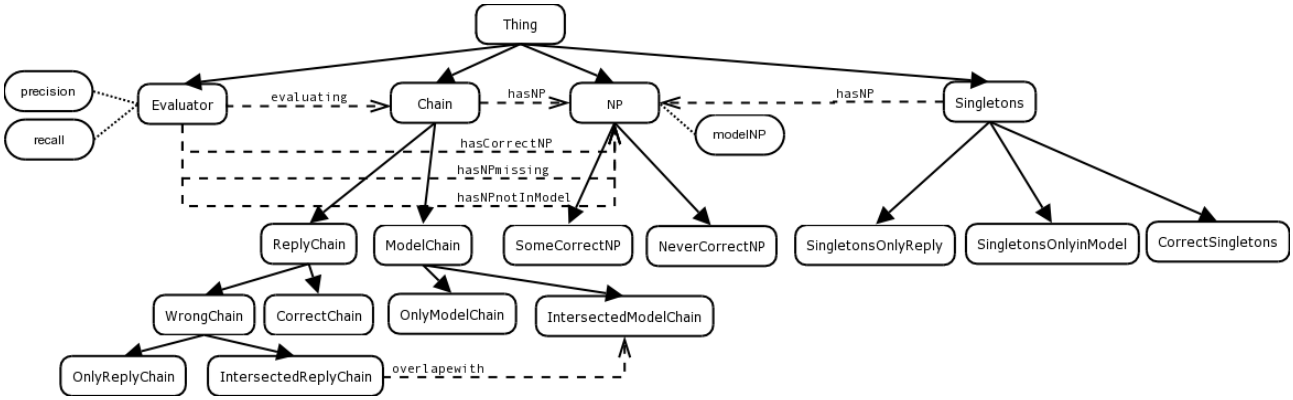

Fig. 1: *Visualization ontology for coreference error detection and analysis*

3.1 Existing Approaches

Little previous work exists on the visualization of coreference resolution results. However, most modern NLP development environments come with graphical user interfaces that are capable of displaying coreference chains as text overlays, e.g., by highlighting or drawing links between entities within a chain. This is also the only approach to coreference visualization discussed in the literature, e.g., within the GATE architecture [1], MMAX [5], or CorefDraw [2].

The main drawback of this approach is that only a part of a coreference chain—for the document text visible within the screen estate—can be viewed. Analyzing larger documents, or cross-document chains, requires permanent scrolling to cover the complete chain, which significantly slows down a developer attempting to gain an overview of all instances within a chain. Moreover, although several chains can potentially be visualized in parallel using e.g. different colors, this quickly becomes too visually complex to be useful.

None of the approaches in the literature suggest task-specific visualization strategies as we defined above.

3.2 Standardized Data Formats

We now review two standardized data formats that are expressive enough for visualizing coreference chains, *Topic Maps* and *OWL Ontologies*.

3.2.1 Topic Maps

Topic Maps are an ISO standard[2] for representing knowledge. They have been designed with a particular emphasis on the findability of information, which makes them a promising target for coreference data.

A Topic Map represents information using *topics, associations,* and *occurrences*. A *topic* is a concept to represent any kind of entity, like a person or organization. *Associations* define the relationships between topics, while *occurrences* link topics with relevant information resources. Each of these three belong to a certain *Topic Type*, which in turn is a topic itself.

Topic Maps are stored and exchanged in an XML-based data format, XTM (XML Topic Maps).

Tool Support. TM4J[3] is an open source topic map engine implemented in *Java*. It includes the graphical browser TMNav, which can display Topic Maps using different rendering pipelines, included a Swing-based and TouchGraph-based one.

3.2.2 OWL Ontologies

Ontologies are a standard technique for representing domain knowledge, and expressive enough to model our domain of discourse, coreference chains. Formal ontologies based on description logics (DL) have been standardized by the W3C in form of the *Web Ontology Language* (OWL) [6].

Tool Support. GrOWL[4] is a visualization and editing tool for OWL. It has been specifically designed for visualizing large ontologies, by allowing a dynamic navigation showing a configurable amount of local context around a node (ABox or TBox). Other tools supporting OWL ontology browsing include Protégé[5] and SWOOP.[6]

3.2.3 Discussion

Both formats have their strength and weaknesses when applied to coreference visualization. Topic Maps are a well-established format and nowadays supported by a whole range of mature visualization tools. In addition, they are easy to generate due to their simple structure. However, the simplicity is also their major downside, as more complex use cases cannot be directly represented using Topic Maps, as we will see below.

Ontologies in OWL-DL format, on the other hand, are much more expressive than Topic Maps, allowing to model complex use cases like coreference evaluation and coreference cluster-based summarization. But the structures expressible in OWL have been designed for machine readability rather than easy visualization for human users. However, the number of robust and scalable ontology visualization tools is steadily increasing, allowing us to upgrade our coreference visualization as they become available.

[2] Topic Maps standard ISO/IEC 13250:2003

[3] Topic Maps For Java, http://tm4j.org/

[4] GrOWL, http://ecoinformatics.uvm.edu/dmaps/growl/

[5] Protégé ontology editor, http://protege.stanford.edu/

[6] SWOOP Hypermedia OWL Editor/Browser, http://www.mindswap.org/2004/SWOOP/

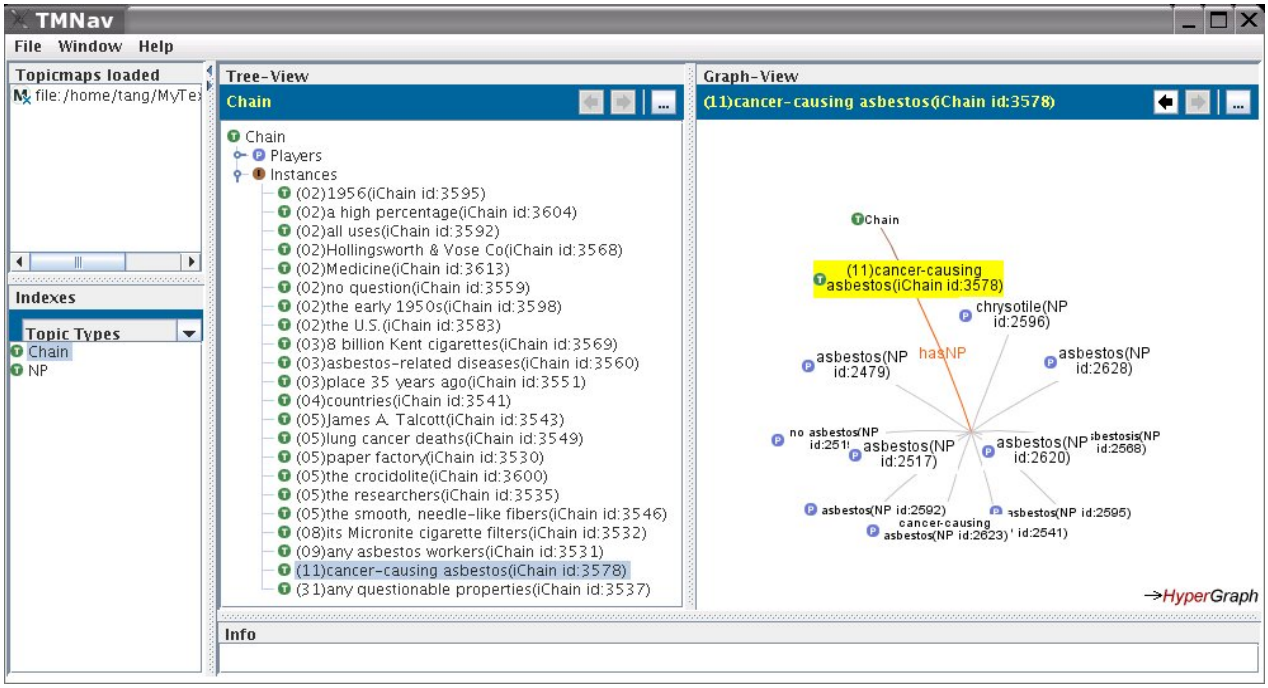

Fig. 2: *Visualization of single-document coreference chains as a Topic Map using a HyperGraph renderer*

4 Coreference Visualization

In this section, we show how to transform coreference chains into the two data formats discussed above, Topic Maps and OWL Ontologies.

We do not assume a particular data format for coreference chains. Within our visualization system, a coreference chain has a unique ID and is represented by a set of noun phrase (NP)[7] ID numbers. Each NP, referenced by its ID, holds meta information like position in the document (start/end) and containing document URI. This representation can be easily created from other formats, like the more implicit MUC style (ID/REF slots on NPs).

4.1 Chain and Document Visualization

Our first use case is to provide a visualization for all coreference chains within a document (set). We differentiate between *inter-document chains* that hold entities from a single document only, and *cross-document chains* that reference entities from two or more documents.

4.1.1 Topic Maps

Coreference chains are transformed to the Topic Map format in the following way:

Topic Type: Three Topic Types are introduced, *Chain* (coreference chain), *NP* (noun phrase, a chain member), and *hasNP* (NP↔Chain relation).

Topic: Each NP and each coreference chain becomes reified as a topic of its corresponding topic type.

Association: Navigation between chains and their NPs becomes possible through adding an association *is_instance_of_hasNP*, which (unsurprisingly) is an instance of the Topic Type *hasNP*.

For visualizing cross-document chains, additional topic types are added to differentiate *IntraChains* from *InterChains*, as well as *Document* topics. Chains are linked with documents through two additional associations, *isIn* to connect chains with their document, and a *spanning* relation indicating which documents are touched by an inter-document chain.

An example of a generated Topic Map visualized using TMNav can be seen in Fig. 2.

4.1.2 OWL Ontologies

Transformation of coreference chains into OWL ontologies is done in two steps: First, we pre-modeled concepts and their relations in an ontology, which is then populated with instances from a system's results:

Classes: Two main classes (TBoxes) are used, CHAIN for a coreference chain and NP for a noun phrase.

Properties: An object property HASNP with the domain CHAIN and the range NP models the connection between chains and NPs.

Instances: Each coreference chain becomes an instance (ABox) of class CHAIN and each noun phrase an instance of the class NP, adding their relations through the object property HASNP.

Fig. 4 shows a simple example for this, visualizing all coreference chains within a (single) document. Each of the white boxes represents a coreference chain, described with its id, a text label, and the number of

[7] Although within the scope of this paper we only discuss coreferencers analyzing NPs, the visualization is not restricted to this type of entity. Other grammatical (e.g., VG) or semantical entities (e.g., Organizations, Proteins) can be visualized in the same fashion.

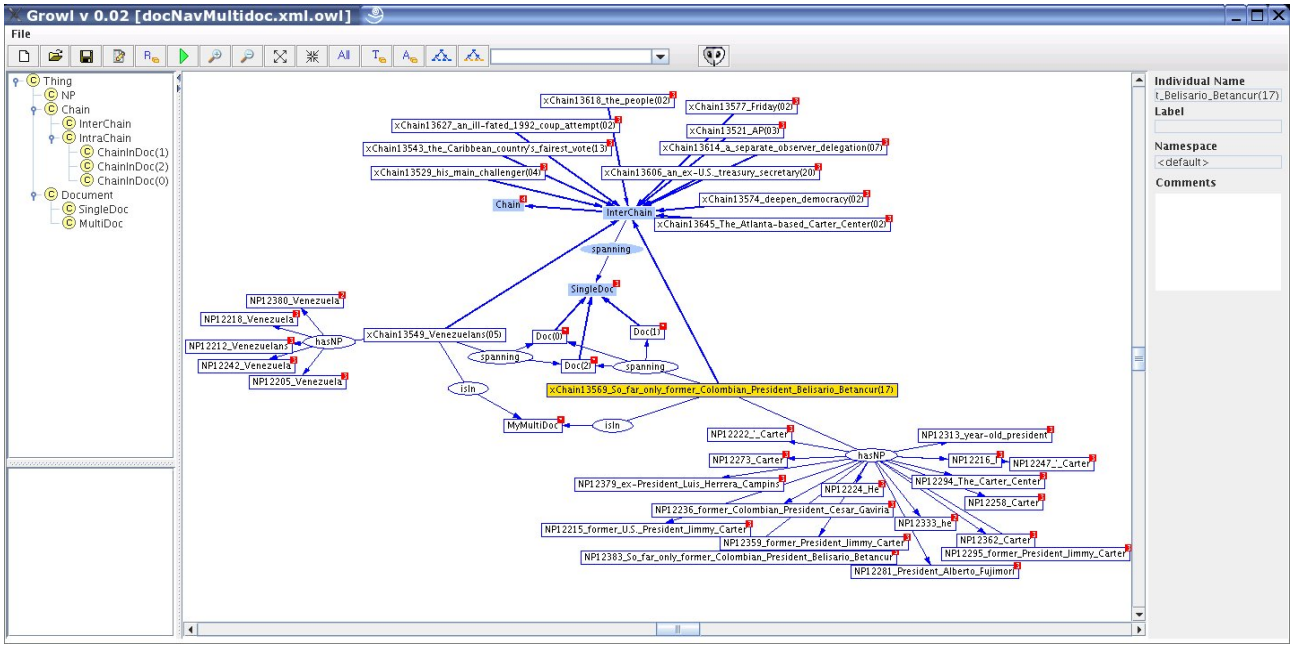

Fig. 3: *Visualization of multi-document coreference chains as an ontology using GrOWL*

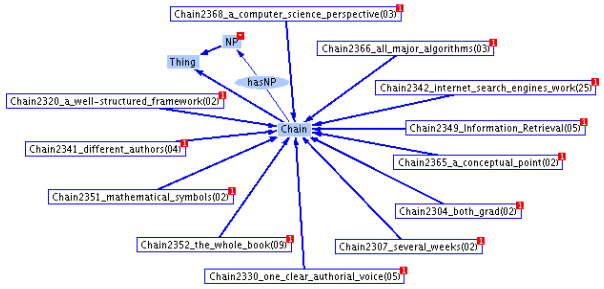

Fig. 4: *Ontology-based visualization using GrOWL showing all coreference chains within a document*

elements. By clicking on one of these boxes, the node can be expanded to show the chain's elements. The user can then navigate to the sentences and documents containing the chain elements. For multi-documents, the ontology is enhanced and further subclassed (see Fig. 1) to represent the connections between documents and intra-/inter-chains (Fig. 3).

4.2 Error Analysis Visualization

For this use case, we only developed an ontology-based visualization, as Topic Maps are not expressive enough to model the complex relationships needed for error analysis, in particular due to the lack of subsumption.

Fig. 1 shows our ontology for error detection and analysis. Besides CHAIN and NP classes we explicitly modeled SINGLETON chains (containing only one NP). In order to analyze coreference results, another entity EVALUATOR is needed that can compute precision/recall values based on a selected metric.[8] Given a manually annotated document as gold standard, we can semantically enrich the visualization ontology with additional information: A CORRECTCHAIN is a chain where a MODELCHAIN and a REPLYCHAIN match exactly. Otherwise, the chain is a WRONGCHAIN that can exhibit several kinds of errors: An INTERSECTEDREPLYCHAIN overlaps with at least one MODELCHAIN (and vice versa), whereas an ONLYREPLYCHAIN does not overlap at all with any MODELCHAIN. Since a reply chain can overlap with multiple model chains, the evaluator computes these subsets for each overlapping model/reply combination.

Furthermore, we can push the semantic annotation for error analysis down to individual noun phrases: If an NP is correct with respect to a particular chain, we can tag it as CORRECTNP. Introducing a relation HASCORRECTNP allows a navigation from an evaluated chain to this class of NPs. Likewise, we can tag all NPs missing in a chain for a quick navigation via a HASNPMISSING relation for a CHAIN. Similarly, we can quickly find superfluous NPs with a HASNPNOTINMODEL relation. We can also differentiate between NPs that have been correctly assigned to at least one reply chain, and NPs that are always wrongly assigned. If a NP has not been assigned to a reply chain for *any* overlapping model chain, we additionally tag it as NEVERCORRECTNP, which is a semantic class of particular interest to a system developer, showing possible serious error sources within a system.

An example for error visualization can be seen in Fig. 5. A user can also navigate starting from the NP class to see all "wrong" NPs, in order to analyze the error case with respect to the various coreference chains computed by the system (missing, additional). Such advanced semantic navigations and visualizations are currently not supported by any other system.

[8] Here, we assume that manually annotated NPs correspond with automatically computed ones. If this is not the case, an additional alignment step has to be introduced, which we do not cover within this paper.

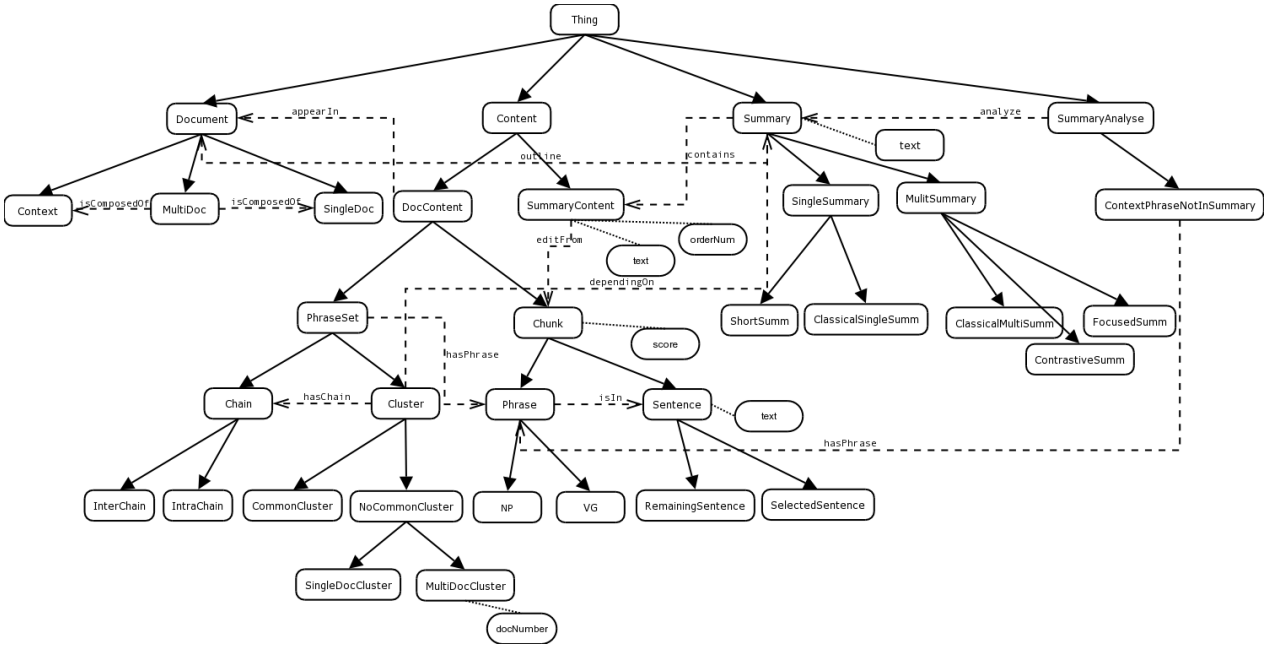

Fig. 6: *Visualization ontology for coreference-based summarization generation analysis*

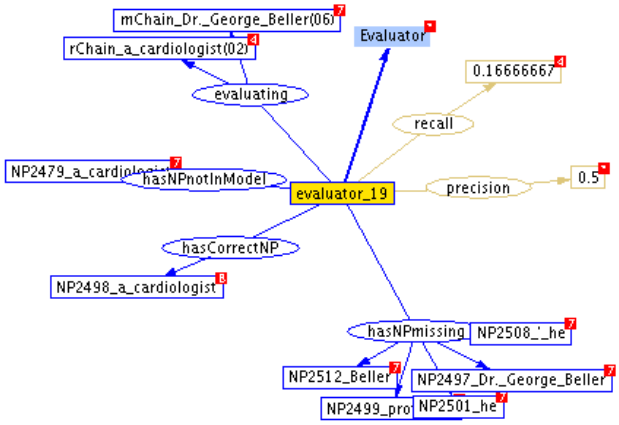

Fig. 5: *Coreference chain error analysis using an ontology visualized in GrOWL*

5 Summarization Visualization

We also investigated the visualization of automatic *summaries* that have been created based on coreference chains and (multi-document) coreference clusters [10]. For an NLP engineer, finding the interrelationships between generated summaries and their underlying coreference chains is another important task during system development. In our approach, the size of coreference chains and clusters determines the important topics in a document set. Sentences are then extracted and assembled to a summary based on these data structures. When performing a qualitative analysis of a generated summary, it becomes necessary to navigate between entities, chains, and sentences in a summary, analyzing their relationships on order to determine *how* and *why* a particular summary was generated.

By enhancing the ontology shown in Fig. 1 with classes for summaries and their constituents (sentences, NPs, etc., see Fig. 6), we can generate visualizations for different kinds of summaries, including single- and multi-document, focused vs. generic, update, and contrastive summaries [11].

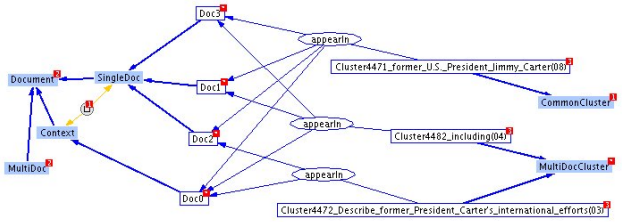

Fig. 7: *Visualizing Coreference Chains and Clusters for the Analysis of a Focused Summary*

An example is shown in Fig. 7, where coreference clusters (sets of coreference chains) are displayed together with the documents they are spanning. Here, the *context* used for generating the focused summary is shown (Doc0) together with the clusters overlapping with both the context and entities in the various documents. In this example, the engineer starts with the context given for the generation of the focused summary (stored in Doc0) and then examines clusters overlapping with the context. These cluster nodes can then be further expanded to display their related entities, including chains, NPs, and sentences selected for a summary.

6 Evaluation

We performed a preliminary evaluation of our work. To determine the impact of the coreference visualization when compared to a text-based output of the chains, we defined a number of tasks typically performed by an NLP engineer during the development and testing of a coreference algorithm. Here, we measure the time it takes a developer to identify certain information (e.g., NPs wrongly assigned to a coreference

chain) with and without our visualizations, in order to assess the impact of our approach. After filtering the resolution results for singletons, the developer had to perform the following tasks:

Task 1: Find all correctly computed chains (i.e., reply identical to model chains)

Task 2: Find all partially correct reply chains

Task 3: Find all reply chains that are completely wrong (i.e., no overlap with any model chain)

Task 4: For each partially correct chain from Task 2, identify the missing/superfluous entities with respect to each overlapping model chain

Task 5: Identify all those entities that are never correctly resolved (i.e., not part of any (partially) correct reply chain)

These tasks have been performed on a newspaper text containing 140 words, resulting in 54 entities. The manually annotated gold standard contains 44 chains, whereas our resolution system [9] computed 38 chains. Then, one of our group's language engineers performed the defined tasks both using the plain system output and the developed visualization system. As can be seen in Table 1, the speedup for solving these tasks based on the visualization offers a dramatic improvement when compared to a text-based output.

	Manual	Visual
0: Remove singletons	4:26 min	n/a
1: Correct chains	1:45 min	10s
2: Partially correct chains	2:22 min	10s
3: Completely wrong chains	1:15 min	10s
4: Missing/superfluous chains	6:56 min	50s
5: Entities incorrect for all chains	16:14 min	10s

Table 1: *Evaluation results comparing tasks performed on a text-based vs. the visualized output*

Of course, it would be possible to develop a custom text-based output format for each of these specific tasks. The important point is that our visualization offers a single, interconnected representation to navigate the result space, allowing the NLP developer to dynamically analyze the results of a coreference algorithm from different perspectives.[9]

7 Conclusions and Future Work

In this paper, we presented two novel ideas for the visualization of coreference chains: (1) A task-centric approach that focuses on use cases of importance to an NLP system developer and (2) Visualization through transforming coreferences into external, standardized data formats supported by existing graphical interfaces. Our implementation demonstrated the feasibility of this approach. The preliminary evaluation results (as well as the practical experiences in our lab) show a dramatic improvement in analysis capabilities compared to state-of-the-art representations of coreference chains.

[9] See, e.g., [3]: *"Compared with an informationally-equivalent textual description of an information a diagram may allow users to avoid having to explicitly compute information because users can extract information 'at a glance'."*

The ideas stated here can also be applied to other areas in NLP, where complex structures are generated by analysis components, as our visualization extension to coreference-based summarization demonstrates. A major advantage of our approach is that language technology engineers can focus on building a conceptual model of the application domain and do not need to invest time in building the graphical renderings themselves. In addition to visualization, OWL-DL ontologies are also supported by powerful querying and reasoning tools, which provides for a completely new paradigm for the analysis of NLP results. When employed together with task-specific semantic visualizations, we expect a major impact on the productivity within the NLP development lifecycle.

References

[1] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proc. of the 40th Anniversary Meeting of the ACL*, 2002.

[2] S. M. Harabagiu, R. C. Bunescu, and S. Trausan-Matu. CorefDraw—A Tool for Annotation and Visualization of Coreference Data. In *ICTAI*, pages 273–279, 2001.

[3] T. Keller and S.-O. Tergan. Visualizing Knowledge and Information: An Introduction. In *Knowledge and Information Visualization*, pages 1–23, 2005.

[4] X. Luo. On Coreference Resolution Performance Metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver, British Columbia, Canada, October 2005. ACL.

[5] C. Müller and M. Strube. A Tool for Multi-Level Annotation of Language Data. In *Proceedings of the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Seattle, USA, 2001.

[6] M. K. Smith, C. Welty, and D. L. McGuinness, editors. *OWL Web Ontology Language Guide*. World Wide Web Consortium, 2004.

[7] M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. A model-theoretic coreference scoring scheme. In *MUC6 '95: Proc. of the 6th conf. on Message understanding*, pages 45–52. ACL, 1995.

[8] C. Ware. Visual Queries: The Foundation of Visual Thinking. In *Knowledge and Information Visualization*, pages 27–35, 2005.

[9] R. Witte and S. Bergler. Fuzzy Coreference Resolution for Summarization. In *Proceedings of 2003 International Symposium on Reference Resolution and Its Applications to Question Answering and Summarization (ARQAS)*, pages 43–50, Venice, Italy, June 23–24 2003. Università Ca' Foscari.

[10] R. Witte and S. Bergler. Fuzzy Clustering for Topic Analysis and Summarization of Document Collections. In Z. Kobti and D. Wu, editors, *Proc. of the 20th Canadian Conference on Artificial Intelligence (Canadian A.I. 2007)*, Springer LNAI 4509, pages 476–488, Montréal, Québec, Canada, May 28–30 2007.

[11] R. Witte and S. Bergler. Next-Generation Summarization: Contrastive, Focused, and Update Summaries. In *Proc. of Recent Advances in Natural Language Processing (RANLP-2007)*, Borovets, Bulgaria, September 27–29 2007.