A Conceptual Framework of Online Natural Language Processing Pipeline Application

Chunqi Shi, Marc Verhagen, James Pustejovsky
Brandeis University
Waltham, United States
{shicq, jamesp, marc}@cs.brandeis.edu

Abstract

This paper describes a conceptual framework that enables online NLP pipelined applications to solve various interoperability issues and data exchange problems between tools and platforms; e.g., tokenizers and part-of-speech taggers from GATE, UIMA, or other platforms. We propose a restful wrapping solution, which allows for universal resource identification for data management, a unified interface for data exchange, and a light-weight serialization for data visualization. In addition, we propose a semantic mapping-based pipeline composition, which allows experts to interactively exchange data between heterogeneous components.

1 Introduction

The recent work on open infrastructures for human language technology (HLT) research and development has stressed the important role that interoperability should play in developing Natural Language Processing (NLP) pipelines. For example, GATE (Cunningham et al., 2002), UIMA (Ferrucci and Lally, 2004), and NLTK (Loper and Bird, 2002) all allow integrating components from different categories based on common XML, or object-based (e.g., Java or Python) data presentation. The major categories of components included in these capabilities include: Sentence Splitter, Phrase Chunker, Tokenizer, Part-of-Speech (POS) Tagger, Shallow Parser, Name Entity Recognizer (NER), Coreference Solution, etc. Pipelined NLP applications can be built by composing several components; for example, a text analysis application such as "relationship analysis from medical records" can be composed by Sentence Splitter, Tokenizer, POS Tagger, NER, and Coreference Resolution components.

In addition to interoperability, the very availability of a component can also play an important role in building online application based on distributed components, especially in tasks such as online testing and judging new NLP techniques by comparing to existing components. For example, the Language Grid (Ishida, 2006) addresses issues relating to accessing components from different locations or providers based on Service-Oriented Architecture (SOAs) models. In this paper, we explore structural, conceptual interoperability, and availability issues, and provide a conceptual framework for building online pipelined NLP applications.

The conventional view of structural interoperability is that a common set of data formats and communication protocols should be specified by considering data management, data exchange, and data visualization issues. Data management determines how to access, store and locate sources of data. For example, GATE provides pluggable document readers or writers and XML (with meta-data configuration) serialization of reusable objected-based data. UIMA provides document or database readers and writers and XMI serialization of common object-based data structures. The Language Grid provides Java object serialization of data collections. Data exchange strategies describe how components communicate their data. For example, GATE provides CREOLE (Collection of REusable Objects for Language Engineering) data collections for data exchange. UIMA provides CAS (Common Analysis Structure), and NLTK provides API modules for each component type. Similarly, the Language Grid provides LSI

(Language Service Interface) for a concrete ontology for a given language infrastructure. Data visualization facilitates manual reading, editing and adjudication. For example, GATE and UIMA provide XML-based viewers for selection, searching, matching and comparison functionality.

The conventional view of conceptual interoperability is that expert knowledge should be used in bridging heterogeneous components. For example, GATE provides integration plugins for UIMA, OpenNLP, and Stanford NLP, where experts have already engineered the specific knowledge on conversion strategies among these components. This leaves open the question of how one would ensure the interoperable pipelining of new or never-before-seen heterogeneous components, for which experts have not encoded bridge protocols.

In order to achieve an open infrastructure of online pipelined applications, we will argue two points regarding the conceptual design, considering both interoperability and availability:

- Universal resource identification, a SQL-like data management, and a light-weight data serialization should be added with structural interoperability in online infrastructure of distributed components.

- By verifying and modifying inconsistent ontology mappings, experts can interactively learn conceptual interoperability for online heterogeneous components pipelines.

2 Data, Tool and Knowledge Types

Interoperability in building pipelined NLP applications is intended ensure the exchange of information between the different NLP tools. For this purpose, existing infrastructures like GATE or UIMA have paid a lot of attention to common entity based data exchanges between the tools. When exchanging data between heterogeneous tools (e.g., the GATE tokenizer pipelined with the NLTK POS tagger), the knowledge of how these different entity based NLP tools can work together becomes much more important, because there might be exchange problems between heterogeneous data or tool information, and we may need specific knowledge to fix them. Thus, when considering interoperability, the main flow of information should be exchanged in the open infrastructure consisting of source data information, NLP tools information, and the knowledge that allows the tools to work together.

What are the main entity types of data and tools in designing an open infrastructure for online NLP pipeline applications? From an abstract view of how linguistic analysis is related to human knowledge, there are the following: Morphological, Lexical, Syntactic, Semantic, Pragmatic tool classifications; and Utterance, Phoneme, Morpheme, Token, Syntactic Structure, Semantic Interpretation, and Pragmatic Interpretation data classifications. (Manaris, 1998; Pustejovsky and Stubbs, 2013). From a concrete application perspective, where tools are available for concrete text mining for communities such as OpenNLP, Stanford CoreNLP and NLTK, there are classification tools such as Sentence Splitter, Tokenizer, POS Tagger, Phrase Chunker, Shallow Parser, NER, Lemmatizer, Coreference; and data classifications such as Document, Sentence, Annotation, and Feature (Cunningham et al., 2002).

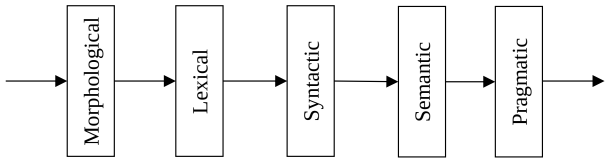


Figure 1: A NLP pipeline can be a (sub)-process of an abstract five-step process

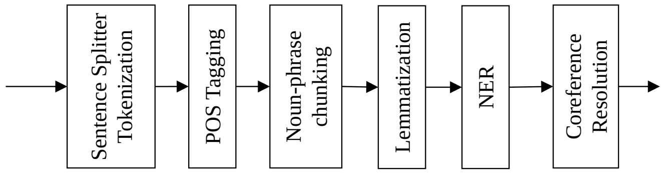


Figure 2: An example NLP pipeline of a concrete six-step process

The knowledge types needed for designing an open infrastructure also can be seen abstractly or concretely. Abstractly, an NLP pipeline should be part of the process of morphological, lexical, syntactic, semantic to pragmatic processing (see Figure 1). From a concrete view, each component of an NLP pipeline should have any requisite preprocessing. For example, tokenization is required preprocessing for POS tagging (see Figure 2). Such knowledge for building NLP pipelines can be interactively determined by the NLP expert or preset as built-in pipeline models.

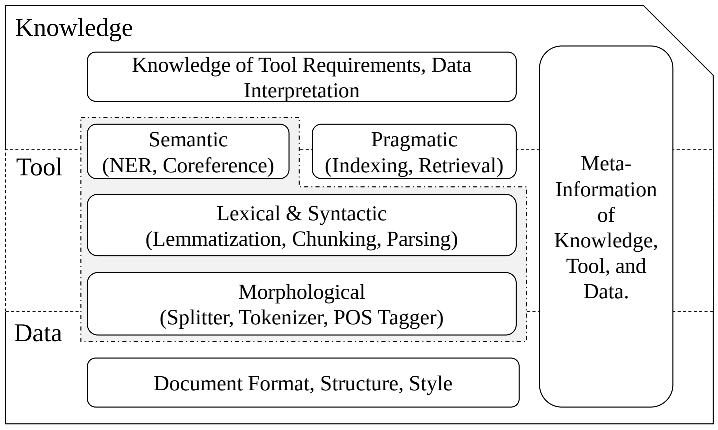

Knowledge
Knowledge of Tool Requirements, Data Interpretation
Tool
Semantic (NER, Coreference)
Pragmatic (Indexing, Retrieval)
Lexical & Syntactic (Lemmatization, Chunking, Parsing)
Morphological (Splitter, Tokenizer, POS Tagger)
Data
Document Format, Structure, Style
Meta-Information of Knowledge, Tool, and Data.

Figure 3: Information for NLP pipeline application description

We can put the above analyzed data, tool, and knowledge types with their meta-information together as the information required for describing an NLP pipeline application (see Figure 3). Regarding the document format, structure and style, for example, the Text Encoding Initiative (TEI)[1] provides one standard for text encoding and interchange, which also enables meta-information description. Concerning the main part (see dashdotted-line part of Figure 3), it is generally referred to as the model of **annotation**. For example, GATE has its own single unified model of annotation, which is organized in annotation graphs. The arcs in the graph have a start node and an end node, an identifier, a type and a set of features (Bontcheva et al., 2004). One standardization effort (Ide and Romary, 2004), the Linguistic Annotation Framework (LAF) architecture is designed so that a pivot format, such as GrAF (Ide and Suderman, 2007), can bridge various annotation collections. Another standardization effort, the Syntactic Annotation Framework (SynAF) (Declerck, 2006), has evolved into the Morpho-syntactic annotation framework (MAF) (Declerck, 2008), which is based on the TEI and designed as the XML serialization for morpho-syntactic annotations. A NLP processing middleware, the Heart of Gold, treats XML standoff annotations for natively XML support, and provides XSLT-based online integration mechanism of various annotation collections (Schäfer, 2006). The UIMA specifies a UML-based data model of annotation, which also has a unified XML serialization (Hahn et al., 2007). Differently from Heart of Gold's XSLT-based mechanism, the conversion tools that bridge GATE annotation and UIMA annotation use GrAF as a pivot and are provided as GATE plugins and UIMA modules (Ide and Suderman, 2009).

Thus, while a pivot standard annotation model like GrAF seems very promising, popular annotation models like those provided by GATE annotations (see Figure 4) or UIMA annotations (see Figure 4) will continue to exist and evolve for a long time. As a result, more bridge strategies, like the conversion plugin (module) of GATE (UIMA) and the XSLT-based middleware mechanism, will continue to be necessary. In the following sections, we consider the issue of the continuing availability of such conversion functions, and whether the current realization of those two conversion strategies is sufficient to bridge the various annotations made available by linguistic experts, without further substantial engineering work.

3 Towards A Conceptual Design of Online Infrastructure

In this section, we discuss the conceptual design of online infrastructure, focusing on both the interoperability and availability of the tools. Concerning the latter, the Service-oriented architecture (SOA) is

[1]http://www.tei-c.org/

```
<!-- GATE -->
<GateDocument>
<TextWithNodes>
<Node id="15"/>Sonnet<Node id="21"/>
</TextWithNodes>
<AnnotationSet>
<Annotation Id="18" Type="Token"
StartNode="15" EndNode="21">
<Feature>
 <Name className="java.lang.String">length</Name>
 <Value className="java.lang.String">6</Value>
</Feature>
<Feature>
 <Name className="java.lang.String">category</Name>
 <Value className="java.lang.String">NNP</Value>
</Feature>
<Feature>
 <Name className="java.lang.String">kind</Name>
 <Value className="java.lang.String">word</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">string</Name>
  <Value className="java.lang.String">Sonnet</Value>
</Feature>
</Annotation>
</AnnotationSet>
</GateDocument>
```

```
<!-- UIMA -->
<xmi:XMI
xmlns:xmi="http://www.omg.org/XMI"
xmlns:opennlp=
"http:///org/apache/uima/examples/opennlp.ecore"
xmlns:cas="http:///uima/cas.ecore"
xmi:version="2.0">

<cas:Sofa
xmlns:cas="http:///uima/cas.ecore"
xmi:id="1" sofaNum="1" sofaID="_InitialView"
mimetype="text"
sofaString="Sonnet." />

<opennlp:Token
xmi:id="18" sofa="1"
begin="0" end="6"
posTag="NNP" />

<cas:View sofa="1"
members="18"/>

</xmi:XMI>
```

Figure 4: Examples of GATE XML annotation and UIMA XML annotation

a promising approach. For example, while the Language Grid infrastructure makes NLP tools highly available (Ishida, 2006), it can still have limitations regarding interoperability issues. Generally, service interfaces can be either *operation-oriented* which allows flexible operations with simple input/output data, or *resource-oriented* which allows flexible input/output data with simple operations. The NLP processing services of Language Grid are more or less *operation-oriented*, and lack a certain structural flexibility for composing with each other. We present a *resource-oriented* view of NLP tools, which should have universal resource identification for distributed reference, an SQL-like data management, and a light-weight data serialization for online visualization. We propose Restful wrapping both data and tools into Web services for this purpose.

Restful wrapping makes both data and tools easy-to-access and with a unified interface, enabling structural interoperability between heterogeneous tools, assuming standoff annotation from various NLP tools is applied. For example, if the NLP tools are wrapped into Restful services so that they are operated through HTTP GET protocol, and the XML serialization of UIMA annotation is applied for input and output, each NLP components will have the same interface and data structure.

Once an internationalized resource identifier (IRI) is given, all the input and output of tools can be distributed and ubiquitously identified. Moreover, a PUT/GET/POST/DELETE protocol of restful data management is equivalent to an SQL-like CRUD data management interface. For example, an IRI can be defined by a location identifier and the URL of the data service (Wright, 2014).

In addition, a lightweight serialization of stand-off annotation can benefit the online visualization of data, which will be easy for experts to read, judge, or edit. For example, the XML serialization of UIMA annotation can be transferred into JSON serialization, which is preferred for online reading or editing.

NLP tool services will be available by applying restful wrapping (see Figure 5). However, structural interoperability based on the restful wrapping is not enough for conceptual interoperability. For example, if an OpenNLP tokenizer is wrapped using HTTP GET protocol and GATE annotation, but a Stanford NLP POS tagger is wrapped using UIMA annotation, it will raise conceptual interoperability issues. Based on the previously mentioned bridging strategies, a conversion service from GATE annotation to UIMA annotation should work, or a transformation interaction with a XSLT-like service should work. We would like to assume that the interaction and contribution of linguistic experts without online support by engineers can solve this issue. But how can we design the interaction to take advantage of such expert knowledge?

We present a semantic mapping-based composer for building an NLP pipeline application (see Fig-

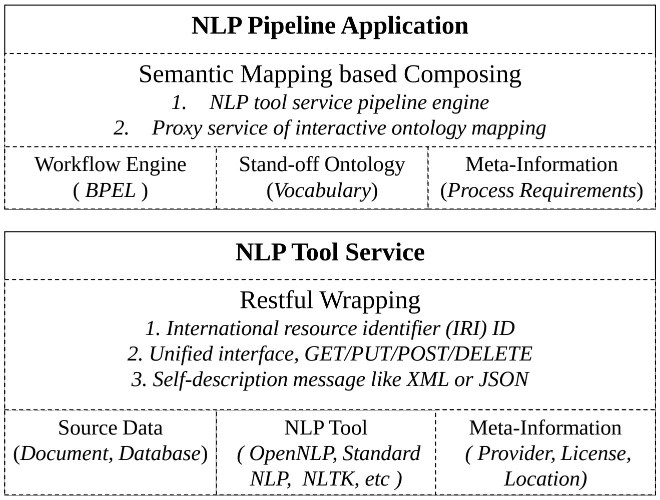
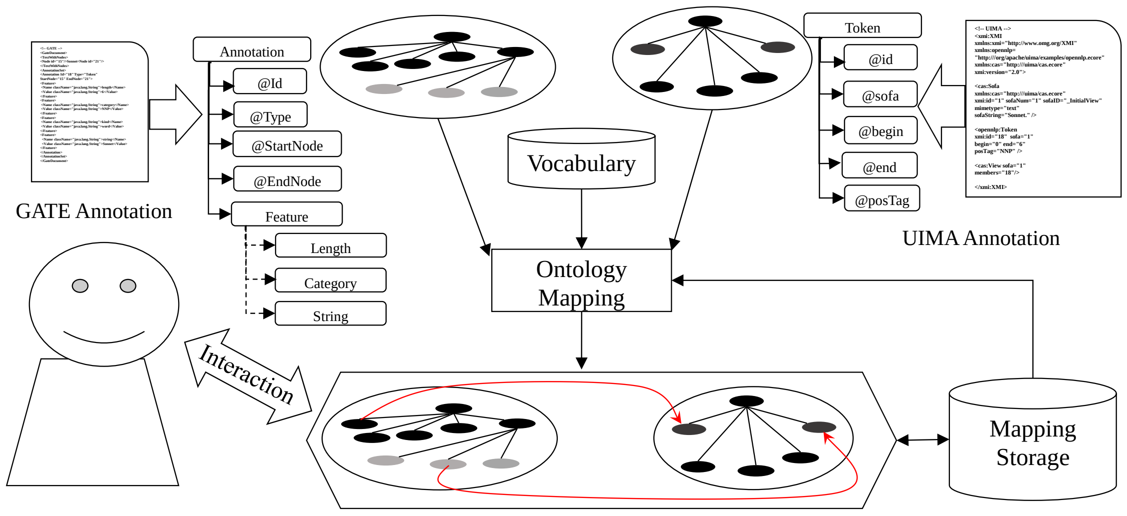

NLP Pipeline Application
Semantic Mapping based Composing
1. NLP tool service pipeline engine
2. Proxy service of interactive ontology mapping
Workflow Engine (BPEL)
Stand-off Ontology (Vocabulary)
Meta-Information (Process Requirements)
NLP Tool Service
Restful Wrapping
1. International resource identifier (IRI) ID
2. Unified interface, GET/PUT/POST/DELETE
3. Self-description message like XML or JSON
Source Data (Document, Database)
NLP Tool (OpenNLP, Standard NLP, NLTK, etc)
Meta-Information (Provider, License, Location)

Figure 5: Conceptual design of online NLP pipeline application

ure 5). Conceptual interoperability requires the same vocabularies for the same concept of a standoff annotation. Once we have the standoff ontology of annotation, we can perform automatic semantic mapping from NLP tool output to that ontology. The interaction from experts will be triggered once the automatic semantic mapping has failed (see Figure 6). For example, both GATE and UIMA XML annotations could be transformed into JSON formation, which is easy to present as tree structure entities. Based on these tree structure entities, automatic ontology mapping tools like UFOme, which identifies correspondences among entities in different ontologies (Pirró and Talia, 2010), can be applied to build up various mapping solutions. Knowledge from experts can also be applied interactively, and successful mapping solutions can be stored for further reference and use.


```
<!-- GATE -->
<GateDocument>
<TextWithNodes>
<Node id="15"/>Sonnet<Node id="21"/>
</TextWithNodes>
<AnnotationSet>
<Annotation Id="18" Type="Token"
StartNode="15" EndNode="21">
<Feature>
 <Name className="java.lang.String">length</Name>
 <Value className="java.lang.String">6</Value>
</Feature>
<Feature>
 <Name className="java.lang.String">category</Name>
 <Value className="java.lang.String">NNP</Value>
</Feature>
<Feature>
 <Name className="java.lang.String">kind</Name>
 <Value className="java.lang.String">word</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">string</Name>
  <Value className="java.lang.String">Sonnet</Value>
</Feature>
</Annotation>
</AnnotationSet>
</GateDocument>
```
```
<!-- UIMA -->
<xmi:XMI
xmlns:xmi="http://www.omg.org/XMI"
xmlns:opennlp=
"http:///org/apache/uima/examples/opennlp.ecore"
xmlns:cas="http:///uima/cas.ecore"
xmi:version="2.0">

<cas:Sofa
xmlns:cas="http:///uima/cas.ecore"
xmi:id="1" sofaNum="1" sofaID="_InitialView"
mimetype="text"
sofaString="Sonnet." />

<opennlp:Token
xmi:id="18"  sofa="1"
begin="0" end="6"
posTag="NNP" />

<cas:View sofa="1"
members="18"/>

</xmi:XMI>
```

Figure 6: Interactive ontology mapping of two different annotations of NLP tools (Tree structures are learned from XML annotations in Figure 4)

The semantic mapping will be interactively created by the experts, when heterogeneous components with different data models are used in the NLP pipeline created by the end-users, who create the NLP pipeline without consideration of components interoperability. It means that this semi-automatically created semantic mapping separates acquiring the knowledge of tool requirements from end-users and acquiring the knowledge of data interpretation from experts (see Figure 3). For example, the end-users chooses two POS Taggers (OpenNLP and NLTK) and two NER tools (OpenNLP and Stanford NLP) components in the NLP application of "relationship analysis from medical records". When NLTK POS

Tagger output are serialized in to JSON formats but cannot be directly used as the input of Stanford NLP NER component which requires the UIMA annotation, a semantic mapping issue will be automatically created and reported to experts. This NLTK POS Tagger JSON format output will be mapped into the standoff ontology of annotation of POS Tagger. After that, this output will bridge with the UIMA annotation of the Stanford NLP NER. This particular semantic mapping between JSON serialization of a NLTK POS Tagger and the standoff ontology of annotation of POS Tagger, and between the standoff ontology of annotation of POS Tagger and the UIMA annotation of Stanford NLP NER will be reused in the NLP application created by other end-users.

Our conceptual framework does not exclusively rely on the above interoperability design. Our conceptual framework (see Figure 5) should integrate existing knowledge of various annotation frameworks, for example, the alignment knowledge from the Open Annotation models (Verspoor and Livingston, 2012) and the pivot bridge knowledge from the GrAF (Ide and Suderman, 2007) under the Linguistic Annotation Framework (LAF). Thus, existing pivot conversion solutions and XSLT-based middleware solutions can also be applied. Our interactive ontology mapping design provides a more flexible choice for linguistic experts to build up NLP pipeline applications on top of heterogeneous components, without online help from engineers. Below we present varying levels of online NLP applications, according to what kind of extra support would be needed for composing different NLP components:

- Components are interoperable without extra data exchange issues. For example, tools are from the same community (e.g., only using OpenNLP tools).
- Components are interoperable with existing solutions of data exchange issues. For example, tools are from popular communities such as GATE plugins or UIMA modules.
- Components are interoperable with extra knowledge from experts. For example, tools are both from popular communities and personal developments or inner group software.
- Components are interoperable with considerable effort from both experts and engineers. For example, tools are developed under novel ontology designs.

According to these levels, our conceptual framework is targeted at the third level of interoperability issues. Our proposal will generate a ontology mapping storage (see Figure 6), which we hope will contribute to improving a standard annotation ontology.

4 Conclusion

In this paper, we have tried to present a conceptual framework for building online NLP pipeline applications. We have argued that restful wrapping based on the Service-Oriented Architecture and a semantic mapping based pipeline composition benefit both the availability and interoperability of online pipeline applications. By looking at the information surrounding the data, tools, and knowledge needed for NLP components pipelines, we explained how experts can be limited in building online NLP pipeline applications without help from engineers, and our restful wrapping and interactive ontology mapping design can help in such situations. Finally, we have described various levels of support needed for building online NLP pipelines, and we believe that this study can contribute to further online implementations of NLP applications.

Acknowledgements

This work was supported by National Science Foundation grants NSF-ACI 1147944.

References

Kalina Bontcheva, Valentin Tablan, Diana Maynard, and Hamish Cunningham. 2004. Evolving gate to meet new challenges in language engineering. *Nat. Lang. Eng.*, 10(3-4):349–373, September.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*.

Thierry Declerck. 2006. Synaf: Towards a standard for syntactic annotation. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. European Language Resources Association (ELRA).

Thierry Declerck. 2008. A framework for standardized syntactic annotation. In Bente Maegaard Joseph Mariani Jan Odijk Stelios Piperidis Daniel Tapias Nicoletta Calzolari (Conference Chair), Khalid Choukri, editor, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

David Ferrucci and Adam Lally. 2004. Uima: An architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348, September.

Udo Hahn, Ekaterina Buyko, Katrin Tomanek, Scott Piao, John McNaught, Yoshimasa Tsuruoka, and Sophia Ananiadou. 2007. An annotation type system for a data-driven nlp pipeline. In *Proceedings of the Linguistic Annotation Workshop*, LAW '07, pages 33–40, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nancy Ide and Laurent Romary. 2004. International standard for a linguistic annotation framework. *Nat. Lang. Eng.*, 10(3-4):211–225, September.

Nancy Ide and Keith Suderman. 2007. Graf: A graph-based format for linguistic annotations. In *Proceedings of the Linguistic Annotation Workshop*, LAW '07, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nancy Ide and Keith Suderman. 2009. Bridging the gaps: Interoperability for graf, gate, and uima. In *Proceedings of the Third Linguistic Annotation Workshop*, ACL-IJCNLP '09, pages 27–34, Stroudsburg, PA, USA. Association for Computational Linguistics.

T. Ishida. 2006. Language grid: an infrastructure for intercultural collaboration. In *Applications and the Internet, 2006. SAINT 2006. International Symposium on*, pages 5 pp.–100, Jan.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bill Manaris. 1998. Natural language processing: A human-computer interaction perspective.

Giuseppe Pirró and Domenico Talia. 2010. Ufome: An ontology mapping system with strategy prediction capabilities. *Data Knowl. Eng.*, 69(5):444–471, May.

James Pustejovsky and Amber Stubbs. 2013. *Natural language annotation for machine learning*. O'Reilly Media, Sebastopol, CA.

Ulrich Schäfer. 2006. Middleware for creating and combining multi-dimensional nlp markup. In *Proceedings of the 5th Workshop on NLP and XML: Multi-Dimensional Markup in Natural Language Processing*, NLPXML '06, pages 81–84, Stroudsburg, PA, USA. Association for Computational Linguistics.

Karin Verspoor and Kevin Livingston. 2012. Towards adaptation of linguistic annotations to scholarly annotation formalisms on the semantic web. In *Proceedings of the Sixth Linguistic Annotation Workshop*, LAW VI '12, pages 75–84, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jonathan Wright. 2014. Restful annotation and efficient collaboration. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).