SLAT 2.0: Corpus Construction and Annotation Process Management

Dain Kaplan, Ryu Iida, Takenobu Tokunaga

Department of Computer Science
Tokyo Institute of Technology
{dain,rui-i,take}@cl.cs.titech.ac.jp

Abstract

Proper annotation process management is crucial to the construction of corpora, which are indispensable to the data-driven techniques that have come to the forefront in NLP during the last two decades. This research proposes a comprehensive annotation abstraction framework to represent any number of layered annotations, their complex relations, and their proper management in large and growing projects. This research focuses extensively on the layering of corpora, which involves corpus construction in a step-wise manner. Proper user management along with proper annotation definitions also help to reduce common annotator error, a concept inherent to the proposed framework. SLAT 2.0, a web-based annotation tool, is then introduced, which implements this framework.

1. Introduction

Corpus-based approaches have risen in popularity over the last two decades in the field of natural language processing (NLP); in many areas corpus-based approaches equal in performance, or rival traditional rule-based methods. With the increase in computational power and the advancement and ease of use of machine-learning (ML) techniques, it is no wonder that corpus-based approaches continue to gain in popularity. In the late 1990s a sizable textbook (Manning and Schuetze, 1999) was published attesting to their merits, and even a decade before the appearance of research pursuing such techniques was not sparse (Charniak, 1993). Corpus-based approaches also allow for methods to be much more language and domain independent, utilizing machine-learning to train models.

The advent of corpus-based approaches also meant that the creation of corpora was required — corpus-based approaches are obviously impossible without them. The individual creation of custom tools for annotation tasks is a tremendous investment of time and labor, which when viewed in a wholistic manner shows how repetitive and wasteful such an investment can be. These custom fitted tools do of course address the demands of the project for which they were born, but also because of this tend towards inflexibility and disposability. Regardless of the problems of interoperability that arise when different tools are used with different data formats (something desirable when trying to cross-test methods with different pre-existing datasets), a cycle forms of creating new tools for trivial tasks, or perhaps even worse conforming and composing annotation criteria to fit within the limits of an existing tool made for a previous project. Corpus-based methods will only grew in scale and complexity, and so it is crucial the annotation tool does not stand in the way.

The bottom line is that corpus creation (and subsequent management) is time consuming enough without having to spend more resources in battling the development of a custom tool. In addition, previous studies have shown that the early phases of a project are the most volatile (Marcus et al., 1993), resulting in rapid changes to the annotation schema. This can also cause delays and cost additional resources if the annotation tool is too rigid or otherwise unable to adapt to the changes. A survey (Dipper et al., 2004) proposed seven categories desirable for any annotation tool: diversity of data, multi-level annotation, diversity of annotation, simplicity, customizability, quality assurance and convertibility. The goal of these categories is to remove the obstacles outlined as problems above. they are well thought out, but there is one caveat to them: they are document-centric, or rather, they simply do not address the bigger scope of managing the annotation process. Furthermore, though they do raise concerns about areas that need to be addressed, they do not propose how these areas are to be tackled. In simple corpora this may not be much of a concern, but in larger projects – and as corpus-based techniques continue to grow and advance so do the sizes of the corpora (Davies, 2009) – it becomes a serious issue. Attempts at diversifying a single corpus with various types of annotations are already on the rise, such as the Discourse TreeBank on top of the Penn TreeBank (Miltsakaki et al., 2004), or the NAIST Text Corpus atop the Kyoto Text Corpus (Iida et al., 2007).

This research proposes a comprehensive annotation abstraction framework that allows for proper annotation process management. The framework is flexible enough to accommodate any annotation scheme, but concrete enough to provide the foundations necessary for an annotation project. Since this research proposes a framework, it could be adopted by any number of annotation platforms as a means for representing the underlying data.

2. Related Work

A plethora of custom annotation tools have been created over the years to meet the specific demands of a project, such as (Stührenberg et al., 2007; Russell et al., 2005); many are created behind the scenes and never see the light of day. They were mostly created because a tool did not exist that was capable of providing the feature set needed for development of the corpus within a reasonable amount of time. A recent example includes Serengeti (Stührenberg et al., 2007), which has been developed for the specific task of anaphoric relations and lexical-chains. Such solutions, however, are not generic; in other words, though they may be quite ideal for one annotation task, they may not readily adapt to other tasks, not to mention the difficulties in data interoperability / interchange. Some general purpose annotation tools have also hit the streets over the years, such as

(Asan and Orăsan, 2003; Cunningham et al., 2002; Dennis et al., 2003; Mueller and Strube, 2001; Callisto, 2002). Some of these are more extensible than others, some are discontinued, some include advanced features for processing data for semi-automatic annotation. PALinkA (Asan and Orăsan, 2003) might be the easiest of these to use, as it requires little setup, but has not seen an update in more than four years and does not support many of the key features we propose here.
None of these focus on multi-tiered corpora, or on annotation process management. As corpus-based techniques continue to flourish, allowing for layered annotations on the same base dataset will be more crucial, as will be managing their interconnectedness; such attempts are on the rise (Miltsakaki et al., 2004; Iida et al., 2007).
There is an existing framework for dealing with annotations called ATLAS (Flexible and Extensible Architecture for Linguistic Annotation) (Bird et al., 2000); but it is highly generalized, and becomes quite complex for the rather simple task of textual annotation (that always involves spanning from one offset in a document to another). Thus we have opted to base our approach on Segments and Links, first introduced by (Takahashi and Inui, 2006; Noguchi et al., 2008), introduced with our extensions in Section 3..

3. Abstracting Annotation

One solution to creating a tool that can be used for multiple tasks (some of which may not yet be conceived), is to abstract the concept of annotation. Using abstraction in this way removes any requirements for the tool to know about the task beforehand. What is necessary, instead, is a definition that specifies what kinds of annotations will be used, and how they relate to one another. If proper constraints are also present in such a configuration, it should allow the user to select only relevant annotation tags which inherently improves the quality of the corpus by forestalling annotator error. Since the system logic is also generalized to handle these configurations, it becomes easier to maintain (there is no custom logic for treatment of specific annotation types). There are other boons to this approach, such as annotation definitions becoming portable.
Figure 1 illustrates this idea visually. In this figure the lowest level of annotation is parts-of-speech. Building upon this, only noun phrases become candidates for selection to annotations when doing named entity annotation. As a third level, annotators are again required to select only annotations from the previous layer as elements in a coreference chain. This sort of layered approach insures higher quality of the resultant corpora, reduces time to creation (targets for selection are greatly reduced so less extraneous user effort is needed) and allows all the layers to use the same data.

3.1. Segments and Links

Segments and Links, first introduced by (Takahashi and Inui, 2006; Noguchi et al., 2008) are the names given two concepts fundamental to abstracting annotations. We expand upon their original definitions as explained below.

Segments
A *segment* in essence is a region of text labeled as a certain type, such as NP or VP (the labeling is arbitrary and up to the user). Because segments are represented as beginning and ending offsets, creating segments that overlap or nest is simple (such as 'diem dulcem habes'). This allows for annotating in layers (there is no interference with one another), which means ~~that a single corpus~~ could be used for multiple tasks without hinderance. However, defining a label alone is not enough, or becomes painstaking at best under certain conditions (such as needing to annotate nouns with different attributes, e.g. singular vs. plural); separate tags could be created for each of these subcategories, but in the case where multiple attributes exist such an approach quickly becomes infeasible. As a solution, our framework allows for the assignment of attributes to segments, and for providing potential attribute-value candidates. This allows you to create an annotation for a "Verb" with possible tenses "past", "present", or "future". There are also cases in which an annotation can only occur within another, such as defining the headword within a noun phrase. Such constraints are also possible.

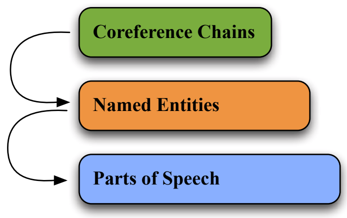

Figure 1: Higher layers access lower layers as a base for more stringent annotation

Links
A *link* is a relationship between two segments. Again, these relations can be defined arbitrarily, so in the sentences "The children played on the freeway; they had a good time." a relation for a semantic role between "children" and "play" or a coreference relation between "children" and "they" is easily obtained. Links can also be transitive and/or directional, useful for hierarchies, coordination, semantic role labeling, etc.
An additional characteristic of links is that they have the ability to limit where they point from and to based on the type and/or the attributes of the target segments. For example, perhaps for a coreference link the source and destination should only be noun phrases with an attribute "NP Type" set to "Entity."

Groups
A *group* is a logical collection of segments, user defined, that encapsulates annotation tasks or roles; the above example of coreference-chains is a good use for this.

4. Framework Overview

Here we present a cursory explanation of the framework. We start general and work our way to specifics. Please refer to Figure 2 for the following explanation.
A *document* is the smallest granular level of content in our framework. It can represent any kind of data you wish (e.g.

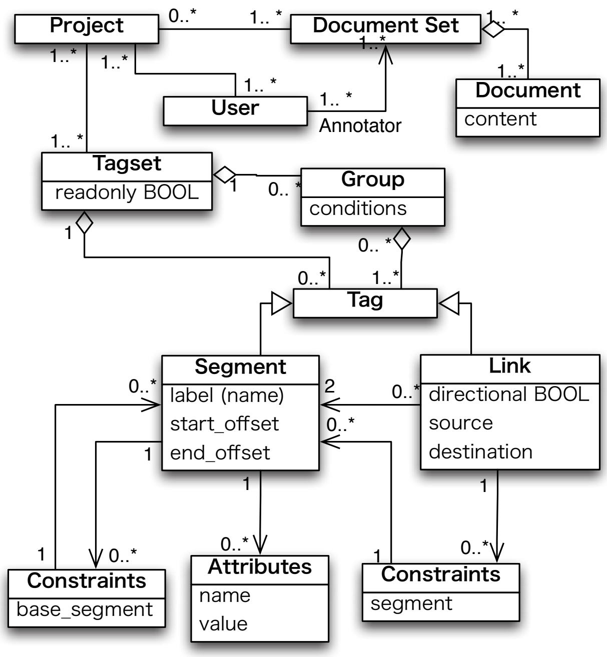

Figure 2: Simplified Relationship Diagram

a news article). Documents can be grouped into *document sets*, which provide a coherent packaging for different documents. A *project* is an annotation task that contains a reference to one or more document sets. Since document sets exist outside of a project, they can be reused for multiple projects, which provides an easy mechanism for layering annotations. We define a *user* as an annotator with access to the system, belonging to one or more projects, and with access to zero or more document sets within any given project. This means that within a project, different users can be assigned to annotate different document sets. Projects are also assigned *tagsets*, where a *tagset* is a collection of *segments*, *links*, and *groups*. These three entities are explained in general in Section 3.1., but a more formal definition now follows. A segment exists in two contexts, one as a predefined label (the *definition*), and two as an *instance* of a defined segment to label a span of text within a document. A segment-definition minimally contains a string label, which becomes at the instance-level a *type* of segment. Segment-definitions can optionally be defined to include binding *constraints*, which require a segment at the instance-level to exist within another segment, such as a Noun existing only within a Noun Phrase. These conditions can optionally contain requirements on values of segment *attributes*, which are explained next. Attributes at the instance-level are name / value pairs representing specific characteristics of an instance. Attributes can be optional or mandatory, and their values can be predefined to limit possible input. Links similarly can be defined to accept only certain segments based on defined-type or attribute constraints. Segment-type constraints can be nested, meaning a different tagset can be used to as a basis for another. This allows us to enforce consistency between different layers of a multi-layered corpus.

5. Annotation Process Management

Next we explain how the proposed framework provides for better annotation process management.
A concrete example best illustrates this. Take a large project like the Penn Treebank (Marcus et al., 1993), which is annotated with parts-of-speech and syntactic structure. This can be represented as a single project, *Project A*, with an annotation tagset for POS-tagging and syntactic structure, *Tagset X*. PropBank (Kingsbury and Palmer, 2002) annotates semantic relations on top of the Penn Treebank. We can represent this with another project, *Project B* and another tagset, *Tagset Y*. Now, this is where management quickly becomes disastrous without the proper system in place. Without any constraints during the annotation process, the wrong tags may be selected by mistake during the second layer of annotation, or worse, data that is not part of a tag at all (a random text span) and should not be selected could be! By properly layering the annotation tagets so that any link in Tagset Y must use segments from Tagset X, and potentially satisfy certain annotation tag attribute values, we can guarantee consistency in Project B (the annotator cannot select an invalid segment). Many implementations simply select the same text spans, but without any constraints. This is a more concrete example of the concept illustrated in Figure 1.
The example illustrates how the proper framework can make layering annotations easy, and improve the quality of the resultant corpora, but what about users? As the size of corpora grow, so do the number of annotators. We have thought about this as well. Though it is only implicitly shown in Figure 1 (including permissions further complicated the diagram), several mechanisms are in place for this. Users and projects can interact in two ways; the first consists of a user configuring project settings, which include managing a document set (adding to / removing from / etc.) and defining what annotation tagsets are applicable to the project's task(s) (and will be used with any document sets). Only administrators should have these privileges. The other type of interaction involves the appearance of the current project when the user is annotating, such as the color of segments / links, when link arrows showing source / destination are visible, etc. As explained above during layered annotation, users may need access to view certain annotation tagsets but only administrators should have access to edit them. In larger projects this separation becomes more important; it safeguards against inconsistencies. Since all this information is encoded within the project's or user's settings, it also means that the data is portable; that conceptually any other platform could read / write to them.
Another advantage of our framework that it enables a single *instance* of a set of documents to be used for multiple projects using different annotation tagsets; the advantages that can be reaped from this are apparent (consistency between all corpora (projects) using the document set), reduced size of storage, etc. Documents may also have links with different source / destination documents, such as pointing back to the initial mention of an entity in a separate news article; the framework allows for this as well.
The biggest merit, however, is that all annotation tagset def-

initions are defined externally by the user, making the system agnostic to them; this means that the behavior is reproducible by any system that can read the configuration files; these files allow for using previous annotation layers as a required basis for subsequent annotation — they encourage it. Such a feature will have an increasing value as various corpus-based methods must be compared to one another on different datasets to testify their merit.

With such a framework in place, the annotation process becomes a matter of simply defining the desired types of annotations, and how they relate to previous layers of already existing annotation. The framework enforces consistency, making the task of expanding your pre-existing corpora never easier than before.

6. SLAT v2.0

SLAT v1.0 provided a web-based interface implementing a basic framework of segments and links. SLAT v2.0, however, will provide a full implementation for proper annotation process management as explained here. It is being developed in PHP and Flash for greater portability and easy access. It can be run on the desktop as an Adobe AIR application with no additional effort, as well. A limited-release beta version of v2.0 will be available soon.

7. Conclusion and Future Work

This paper presents a comprehensive framework for representing the complex structure needed to properly manage the annotation process; specifically it achieves this by abstracting the concept of annotation, and providing a mechanism for layering annotations through use of segment and link types, constraints, and attributes. With this framework in place, the annotation process becomes a simple matter of defining the desired types of annotations (as tagsets), and how they relate to previous layers of al-ready existing annotation, and then annotating! It also promotes maintainability by forcing the user to define annotation tags and their relationships, making the system agnostic to such definitions. Future work includes extending the framework to facilitate comparing annotator agreement, and merging annotations.

Acknowledgments

This work is partially supported by the Grant-in-Aid for Scientific Research Priority Area Program "Japanese Corpus" (2006-2010).

References

Constantin Or Asan and Constantin Orăsan. 2003. Palinka: A highly customisable tool for discourse annotation. In *Proc. of the 4 th SIGdial Workshop on Discourse and Dialogue, ACL '03*, pages 39–43.

Steven Bird, David Day, John S. Garofolo, John Henderson, Christophe Laprun, and Mark Liberman. 2000. Atlas: A flexible and extensible architecture for linguistic annotation. *CoRR*, cs.CL/0007022.

Callisto. 2002. `http://callisto.mitre.org`.

Eugene Charniak. 1993. *Statistical Language Learning*. The MIT Press.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. Gate: A framework and graphical development environment for robust nlp tools and applications. In *Proc. of the 40th Annual Meeting of the ACL*.

Mark Davies. 2009. Contemporary American English (1990-2008+). *International Journal of Corpus Linguistics*, 14(2):159–190.

Glynn Dennis, Brad Sherman, Douglas Hosack, Jun Yang, Wei Gao, H. Clifford Lane, and Richard Lempicki. 2003. David: Database for annotation, visualization, and integrated discovery. *Genome Biology*, 4(5):P3+.

Stefanie Dipper, Michael Götze, and Manfred Stede. 2004. Simple annotation tools for complex annotation tasks: An evaluation. In *Proc. of the LREC Workshop on XML-based Richly Annotated Corpora*.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a japanese text corpus with predicate-argument and coreference relations. In *Proc. of the Linguistic Annotation Workshop*, pages 132–139, Prague, Czech Republic, June. Association for Computational Linguistics.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proc. of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1989–1993.

Christopher D. Manning and Hinrich Schuetze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1 edition, June.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004. The penn discourse treebank. In *Proc. of LREC 2004*.

Christoph Mueller and Michael Strube. 2001. MMAX: A tool for the annotation of multi-modal corpora. In *Proc. of the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 45–50.

Masaki Noguchi, Kenta Miyoshi, Takenobu Tokunaga, Ryu Iida, Mamoru Komachi, and Kentaro Inui. 2008. Multiple purpose annotation using SLAT – segment and link-based annotation tool. *Proc. of 2nd Linguistic Annotation Workshop*, pages 61–64.

B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. 2005. Labelme: A database and web-based tool for image annotation. Technical report, Tech. Rep. MIT-CSAIL-TR-2005-056, Massachusetts Institute of Technology.

Maik Stührenberg, Daniela Goecke, Nils Diewald, Alexander Mehler, and Irene Cramer. 2007. Web-based annotation of anaphoric relations and lexical chains. In *Proc. of the Linguistic Annotation Workshop*, pages 140–147, Prague, Czech Republic, June. Association for Computational Linguistics.

Tetsuro Takahashi and Kentaro Inui. 2006. A multipurpose corpus annotation tool: Tagrin. *Proc. of the 12th Annual Conference on Natural Language Processing*, pages 228–231.