Improving the Effectiveness of Machine-Assisted Annotation

Paul L. Felt
Brigham Young University - Provo

Improving the Effectiveness of Machine-Assisted Annotation

Paul Felt

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

June 2012

ABSTRACT

Improving the Effectiveness of Machine-Assisted Annotation

Paul Felt
Department of Computer Science, BYU
Master of Science

Annotated textual corpora are an essential language resource, facilitating manual search and discovery as well as supporting supervised Natural Language Processing (NLP) techniques designed to accomplishing a variety of useful tasks. However, manual annotation of large textual corpora can be cost-prohibitive, especially for rare and under-resourced languages. For this reason, developers of annotated corpora often attempt to reduce annotation cost by offering annotators various forms of machine assistance intended to increase annotator speed and accuracy.

This thesis contributes to the field of annotated corpus development by providing tools and methodologies for empirically evaluating the effectiveness of machine assistance techniques. This allows developers of annotated corpora to improve annotator efficiency by choosing to employ only machine assistance techniques that make a measurable, positive difference.

We validate our tools and methodologies using a concrete example. First we present CCASH, a platform for machine-assisted online linguistic annotation capable of recording detailed annotator performance statistics. We employ CCASH to collect data detailing the performance of annotators engaged in syriac morphological analysis in the presence of two machine assistance techniques: pre-annotation and correction propagation. We conduct a preliminary analysis of the data using the traditional approach of comparing mean data values. We then demonstrate a Bayesian analysis of the data that yields deeper insights into our data. Pre-annotation is shown to increase annotator accuracy when pre-annotations are at least 60% accurate, and annotator speed when pre-annotations are at least 80% accurate. Correction propagation's effect on accuracy is minor. The Bayesian analysis indicates that correction propagation has a positive effect on annotator speed after accounting for the effects of the particular visual mechanism we employed to implement it.

Keywords: Syriac, Bayesian methods, Annotated Corpora, Machine-Assisted Annotation, Machine Assistance

ACKNOWLEDGMENTS

A large number of people were instrumental in the successful creation of this thesis. I am especially grateful to Eric Ringger and Kevin Seppi for their tireless work in helping me to improve both the ideas contained in this thesis as well as their presentation. Robbie Haertel was also an important contributor to nearly every part of this work, and was particularly key to improving the design of the user study.

Running the user study itself required considerable amounts of work by several groups of people. Kristian Heal graciously donated many hours working with me to improve the Syriac annotation interface and create annotator training material. David Taylor, Carl Griffin, and Aaron Butts carefully worked through a beta user study and provided invaluable feedback. Owen Merkling, Warren Lemmon, Jeremy Sandberg, Earl Greathouse, Benjamin Hansen, and Kevin Black all helped improve CCASH, the software used to run the user study. Finally, the user study would have been impossible if not for the generous time donations of the participants themselves.

Most of all, I need to acknowledge the debt I owe to my wife, Stephanie, for seeing this through with me. Without her encouragement and support, this work would not have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Machine-Assisted Annotation

1.1 Introduction

The current success and widespread use of data-driven techniques for processing human language make annotated corpora an essential language resource. For instance, many popular Natural Language Processing (NLP) algorithms require significant amounts of high quality (often human-annotated) training data in order to perform effectively. Also, annotated text can be useful in its own right as a means of exploring and understanding the text itself. For example, one might use part-of-speech annotations and syntactic dependencies to study the diachronic prominence of ideas in a language.

There is an urgent need to produce more annotated corpora. Because data-driven prediction techniques learn to mimic patterns found in training data, they perform best when the data they learn from are very similar to the data they are to be used upon. For example, an automatic grammatical tagger trained on hand-labeled news articles will likely perform well when used to automatically label similar articles, but may perform poorly on transcribed spontaneous vocal utterances. Therefore corpora must be labeled for each domain of interest. In addition, a growing number of linguistic tasks have been proposed including part of speech tagging, named entity recognition, constituent parsing, and dependency parsing [28]; supertagging and deep grammatical parsing [2]; co-reference resolution [43], sentiment analysis [37], information extraction [10], and many more. Solving high level language problems may involve integrating state-of-the-art solutions for many different NLP tasks, such as IBM's recently publicized Watson project, designed to accomplish deep question answering [17]. Before applying data-driven prediction techniques to a new problem, a corresponding labeled corpus must be constructed for all necessary subtasks.

The number of annotated corpora required to adequately cover the cross product of domains and tasks poses a problem even for highly resourced languages (e.g. English), and is far more daunting for the many under-resourced languages of the world, including a great number of languages that are in the process of disappearing [23]. Producing corpora documenting these endangered languages while there are still living native speakers is an endeavor of great linguistic importance. Because there are insufficient resources to fulfill all of these needs, it is highly desirable to find ways of reducing the cost of creating annotated corpora.

1.2 The Annotation Landscape

This section sketches general approaches that have been taken to improve the process of creating labeled corpora. It also points out some of the strengths and weaknesses to those approaches, and ends by explaining, in context, the scope and significance of the current work.

Many methods have been employed to avoid the costs of a traditional labeling project. A large body of work has built up around the notion of crowd-sourcing, using internet participation on a grand scale to solicit very noisy labels at very low cost using systems like Amazon's Mechanical Turk [26].[1] Others have used cleverly constructed games to elicit labels from participants [47]. Similarly, a labeled corpus may be constructed for free if one can discover data that has been implicitly labeled. For example, Pang et al. [38] crawled an online corpus of movie reviews that were already labeled with a summary numerical rating (e.g. 1 through 5 stars). However, many of these techniques are only effective for highly resourced languages and for labeling tasks which do not require much expertise.

When manual corpus creation is unavoidable, various flavors of machine assistance have been proposed to increase annotator speed and accuracy. Marcus et al. [29] and many others have used pre-annotation, presenting annotators with automatically labeled sentences so that they need

[1] http://mturk.amazon.com

merely correct errors rather than annotate from scratch. Kristjansson et al. [27] proposed correction propagation as an extension to pre-annotation in which multi-part pre-annotatinos are dynamically revised whenever the human annotator corrects an erroneous portion of a pre-annotation. Active learning is another approach that addresses the problem from the machine's point of view. Rather than assisting annotators to work quickly, active learning attempts to assist the automatic labeler to learn quickly by presenting the human annotator with examples that are likely to be of the most value to the automatic labeler [41]. Higher quality pre-annotations may, in turn, reduce annotation cost. In a similar vein, several researchers have explored ways of reducing the cost of training a high quality automatic annotator by allowing experts to inject expert knowledge into the model to accelerate the training process [7, 14, 19, 34].

However, all machine assistance techniques rely on the existence of an automatic helper of some kind, and, as previously noted, most state-of-the-art automatic learners rely on the existence of already labeled data. The result of this circular dependency is that machine assistance tends to be of a poor quality to begin with, slowly improving as labels are accumulated. If the quality is sufficiently poor, it is entirely possible that machine assistance could do more harm than good. For example, pathologically bad pre-annotations might incur a time cost by distracting and antagonizing human annotators. In addition, the best implementation of a given machine assistance technique is not guaranteed to be the same for every linguistic task. For some tasks, presenting the single most likely pre-annotation could be the most effective pre-annotation strategy [29], while for other tasks, it may be more effective to present the top k possibilities [18]. A crucial question, then, regarding machine assistance techniques is determining when and how they ought to be used for each linguistic task and domain.

This thesis contributes to the field of labeled corpus development in a small but important way: by providing a methodology and tools to evaluate the performance of machine assisted annotation. We demonstrate the effectiveness of our tools and methodology by choosing two machine assistance techniques to evaluate in the context of a non-trivial linguistic annotation task. The machine assistance techniques we choose are pre-annotation and correction propagation. The linguistic task we choose is Syriac morphological analysis. As a result, this thesis also make an important contribution to the under-resourced language of Syriac by answering the question of when pre-annotation and correction propagation are effective machine assistance techniques for Syriac morphological analysis. This chapter will proceed to state more clearly the thesis to be proven, then discuss in detail Syriac morphological analysis, pre-annotation, and correction propagation.

1.3 Thesis Statement

Pre-annotation and correction propagation can increase the speed and accuracy of annotators engaged in Syriac morphological annotation.

1.4 Syriac Morphological Analysis

Scholars at the Neal A. Maxwell Institute for Religious Scholarship at BYU and at the Oriental Institute at the University of Oxford are jointly working on a project called the Syriac Electronic Corpus project, with the goal of creating a comprehensive, labeled corpus of classical Syriac. Classical Syriac ('*kthobonoyo*') is an under-resourced Semitic language of the Christian Near East and a dialect of Aramaic. It is currently in use as a liturgical language but was a true spoken language up until the eighth century when it was largely supplanted by Arabic. Many prolific authors wrote in Syriac. The goal of the Syriac Corpus project is to annotate these texts with morphological analyses to facilitate systematic study of Syriac by historians, linguists, and language learners.

ܠܡܰܠܟܟ݂ܽܘܢ	ܡܰܠܟ	ܡܠܟܐ	ܡܠܟ
token	stem	citation form	root

Figure 1.1: The Syriac word token LMaLK'K,uON "to your king" and its related forms

Morphological analysis of Syriac is the process of segmenting a word into its constituent morphemes and labeling each with its grammatical function(s). For our purposes, the primary morpheme is the "stem", namely the remainder of the token after removing morphological suffixes and prefixes. The dictionary citation form (or baseform) and the root are identified from the stem.

In contrast to English, where searching for a few forms of a word or using simple query-expansion is often sufficient for discovering patterns reflecting the word's usage and meaning, in Semitic languages search and discovery are not so straightforward. If we could search Syriac texts on citation forms or even on roots, we could search for and discover patterns as easily as in English; however, Semitic roots are altered significantly by expressive inflectional and derivational morphological processes. Consequently, inflected forms of any given Syriac root are numerous. As a result, searching Syriac text is impaired since one must either limit one's query to a single inflected form or use heuristics to expand the query, buying higher recall at the price of lower precision.

A morphologically annotated digital corpus of a lesser studied language such as Syriac lends itself to search and therefore to careful study in a way that formerly only experts could attempt based on long years of familiarity. Such annotated corpora enable scholars to study and discover the contributions of and trends in historical documents. One outstanding example of such a corpus is the Dead Sea Scrolls Electronic Library (DSSEL), assembled by the Center for the Preservation of Ancient Religious Texts (CPART) in the Neal A. Maxwell Institute for Religious Scholarship at Brigham Young University [46]. The Syriac Corpus will be an artifact of similar value, useful to linguists, Syriac students, and scholars of Syriac, the Near East, and Eastern Christianity.

Unfortunately, creating annotated corpora can be extremely time consuming. The Way International Foundation, a Biblical research, teaching, and fellowship ministry, spent 15 years labeling the Syriac New Testament with morphological annotations [25]. The Syriac New Testament consists of approximately 100,000 words. Similarly, two Syriac scholars recently required 18 months to hand label less than half of the Old Testament [24]. By contrast, the Syriac Corpus aims to encompass approximately 10,000,000 words. To achieve this goal in a timely manner it will be necessary to increase the speed of annotation.

1.5 Pre-annotation and Correction Propagation

Pre-annotation, also called automatic annotation or pre-labeling, is a form of machine assistance that has the potential to reduce overall annotation cost by using NLP algorithms to automatically annotate each instance (i.e. token) before it is presented to an expert annotator. Expert annotators then need only review and correct the proposed annotations, which can be much quicker than annotating from scratch.

Kristjansson et al. [27] describe an enhancement to pre-annotation for annotation tasks that require making multiple decisions (e.g. labeling each word in a sentence) which they call correction propagation. Correction propagation is a technique in which annotator corrections to any part of a multi-part annotation are returned to the machine annotator, allowing the machine annotator to improve its original hypotheses and fix downstream errors in the annotation, potentially saving the annotator the effort of correcting them. Kristjannson et al. give the example of an annotator identifying contact information in free text. In such a case, correcting a pre-annotated given name might allow the automatic annotator to correctly identify the corresponding surname and address.

One problem affecting both pre-annotation and correction propagation is that they require a model capable of supplying or updating automatic annotations. However, as noted earlier, many NLP algorithms for building such a model require already annotated training data. For tasks and languages without already existing resources, one must therefore begin the annotation process with low quality pre-annotations and periodically retrain the automatic annotator as more data is labeled. Although pre-annotation and correction propagation can help, it is conceivable that sufficiently inaccurate predictions could reduce annotator speed or accuracy. Because of this, before building annotated corpora in domains with little labeled data, it would be desirable to have a sense of how accurate a model must be in order to make pre-annotation and correction propagation helpful instead of harmful. This thesis presents the collection and analysis of data detailing the effectiveness of pre-annotation and correction propagation on Syriac morphological analysis.

1.6 Related Work

In order to apply pre-annotation to Syriac morphological analysis, we first require a model capable of generating analysis hypotheses for Syriac text. We use Syromorph, a probabilistic morphological analyzer for Syriac developed by McClanahan et al. [30]. Syromorph accomplishes Syriac morphological analysis as a pipelined sequence of classification and transduction tasks. Each task in the pipeline relies on the data and on the results of all tasks preceding the current task in the pipeline. Syromorph first segments each word into its parts: prefix, stem, and suffix. Syromorph then predicts a baseform, or dictionary citation form, for the stem. Finally, Syromorph predicts the grammatical attributes of the stem and suffix.[2]

In order to apply correction propagation to Syriac morphological analysis, our model must be capable of constraining its predictions to match partial labelings. Kristjansson et al. [27] propose a constrained Viterbi decoding algorithm for linear conditional random field models (CRFs). Because Syromorph is a pipelined sequence model, we are able to adapt Kristjannson's method of constrained decoding, with the difference that we use an n-best beam decoder instead of Viterbi decoding. Other approaches to constrained inference include the work of Chang et al. [8] who use integer linear programming to constrain inference in large class of models, including CRFs.

Pre-annotation has been evaluated on a variety of linguistic annotation tasks. Marcus et al. [29] evaluated pre-annotation using an interface embedded in the GNU Emacs Editor to label the Penn Treebank with English part-of-speech tags to the Penn Treebank. They timed by hand four annotators and reported that pre-annotation more than doubled annotation speed and also increased accuracy and inter-annotator agreement. Chiou et al. [9] timed two annotators and reported a 70% increase in annotation speed using pre-annotation on a Chinese Treebank annotation task. Ganchev et al. [18] used a custom web-based tool to do named entity recognition (NER). In order to make pre-annotation effective for NER, they found they had to apply their pre-annotation approach at

[2] In accordance with the current needs of the Syriac Corpus project, the original Syromorph (v1.0) has been modified slightly so that it no longer predicts a root form (current version is 2.1). The reason for this change is that the ultimate goal of the project is to link each token to a baseform dictionary entry, and the root form comes for free with this linkage.

a finer grain than whole-tree by presenting annotators with a set of plausible guesses instead of a single best guess. They recorded by hand the time of a single annotator and reported a more than 50% increase in speed. Brants and Plaehn [5] applied pre-annotation to parse tree labeling. In order to make pre-annotation effective for parse tree labeling, they found they had to alter their pre-annotation approach by creating an interactive parse tree where annotators accept or reject suggestions starting at the parse tree's leaves and working their way to the root. These results are encouraging, but unfortunately they are not strong enough to let us conclude that pre-annotation will necessarily be effective for Syriac morphological analysis. For one thing, most of the tests involved only one or two annotators. More importantly, pre-annotation had to be adapted before it was effective for some of the tasks, implying that for complex annotation tasks, naïve pre-annotation may not be effective. It is unclear which, if any, of the previous pre-annotation results apply to Syriac morphological analysis or to other linguistic annotation tasks.

Correction propagation has been evaluated on far fewer tasks than pre-annotation. As has already been noted, Kristjansson et al. [27] applied correction propagation to the task of information extraction, interactively assisting simulated users to fill in database fields. They evaluated the performance of correction propagation in simulation and showed that automatic annotator accuracy significantly increased after even a single correction. They also showed that correction propagation significantly reduced the expected number of user interactions with a proposed graphical user interface. These results are promising; however, because of the differences between information extraction and Syriac morphological analysis, previous work is insufficient to conclude that correction propagation will be effective for Syriac morphological analysis.

1.7 Publications Roadmap

The balance of this thesis consists of the published and submitted papers resulting from this project. They describe the work that was required to test the thesis statement: building tools, gathering data with a user study, and analyzing the data. Prepended to each paper is a brief explanation of how that paper fits into the larger context of the thesis. This section provides a brief summary of each paper and how it contributes to the thesis.

- Chapter 2 has been published in the Seventh International Conference on Language Resources and Evaluation [15]. This paper describes the design and architecture of CCASH, a framework that allows collaborative online annotation, facilitates machine assistance, and also records detailed timing information about all user interactions.
- Chapter 3 will be published in the Eighth International Conference on Language Resources and Evaluation [16]. This paper reports on a controlled user study in which nine participants each annotated 30 sentences with Syriac morphological analyses, assisted by pre-annotation and correction propagation. The study design is explained in detail along with a simple analysis of the data in terms of time and accuracy.
- Chapter 4 has been submitted for review to 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP). This paper takes a deeper look at the timing information gathered in the user study using a Bayesian methodology. The Bayesian analysis corraborates previous conclusions and yields additional insights that allow us to improve the way we are applying correction propagation to Syriac morphological analysis.

Bear in mind that the papers in Chapters 2, 3, and 4 are self-contained works, so although their main contributions are different, they necessarily contain a good deal of redundant introductory material. Also, all paper references have been merged with the general list of thesis references.

Chapter 2

CCASH: A Web Application Framework for Efficient, Distributed Language Resource Development

Author List

Paul Felt, Owen Merkling, Marc Carmen, Eric Ringger, Warren Lemmon, Kevin Seppi, Robbie Haertel

Publication Venue

The Seventh International Conference on Language Resources and Evaluation (LREC) 2010

MS Thesis Context

This paper fits into the larger thesis by creating the software tools necessary in order to implement and evaluate machine assistance methods. Before CCASH, no annotation software existed that was designed to record detailed timing information about user interactions.

Abstract

We introduce CCASH (Cost-Conscious Annotation Supervised by Humans), an extensible web application framework for cost-efficient annotation. CCASH provides a framework in which cost-efficient annotation methods such as Active Learning can be explored via user studies and afterwards applied to large annotation projects. CCASH's architecture is described as well as the technologies that it is built on. CCASH allows custom annotation tasks to be built from a growing set

of useful annotation widgets. It also allows annotation methods (such as AL) to be implemented in any language. Being a web application framework, CCASH offers secure centralized data and annotation storage and facilitates collaboration among multiple annotations. By default it records timing information about each annotation and provides facilities for recording custom statistics. The CCASH framework has been used to evaluate a novel annotation strategy presented in a concurrently published paper, and will be used in the future to annotate a large Syriac corpus.

2.1 Introduction

The current success and widespread use of data-driven techniques in language-related fields make annotated corpora an often essential language resource. For instance, many popular Natural Language Processing (NLP) algorithms require significant amounts of human-annotated training data in order to perform effectively. Also, annotated text can be useful in its own right as a means of qualitatively exploring the annotated text. For example, one might use part-of-speech (POS) annotations to study the syntax of a language, or morphological annotations to study the formation of words in a morphologically rich language.

Along with the need for annotated corpora comes the need for tools capable of creating these corpora. However, the process of creating annotated corpora is not trivial. For one thing, employing human specialists to annotate each instance in a corpus by hand can be prohibitively costly. A general purpose annotation tool should make use of existing cost-efficient annotation methods such as automatic annotation and Active Learning (see Section 2.2). However, cost-efficient annotation is an area of active research, so annotation tools should also be sufficiently flexible to encourage novel methods to be implemented and explored. Indeed, since the effectiveness of various annotation methods may vary across tasks and domains, even projects interested only in applying known annotation methods to a large corpus may wish to conduct exploratory studies to compare the efficiency of several annotation methods before proceeding on a large scale. In addition to cost, many other problems must be dealt with. If the annotation task being conducted is uncommon, project developers may need to customize an existing annotation tool or create their own custom tool to implement that annotation task. Annotation projects that employ multiple annotators must solve problems of data distribution and consistency. Such projects must somehow distribute views of the corpus to each annotator and collect annotations into a central location, handling any conflicts among the annotations.

Although this discussion by no means exhausts the demands that might be made of a general-purpose annotation tool, we believe they are an important subset. Ideally then, an annotation tool would offer at a minimum the following high-level features:

- Accommodate proven cost-efficient annotation methods
- Encourage novel cost-efficient annotation methods
- Facilitate exploratory studies and comparisons of annotation methods (e.g. measure annotation costs)
- Accommodate custom annotation tasks
- Coordinate the efforts of multiple annotators

In this paper we introduce CCASH (Cost-Conscious Annotation Supervised by Humans), a web application framework for corpus annotation designed to implement this feature set by using familiar programming paradigms, open standards technologies, and by providing reasonable default implementations whenever possible, always allowing those with unique requirements to define their own features from the ground up.

The remainder of this paper is organized as follows: in Section 2.2 we describe annotation projects, studies, and tools that influenced CCASH's design and implementation. In Section 2.3 we explain our decision to implement CCASH as a web application. In Sections 2.4 and 2.5 we describe CCASH's architecture and implementation details. In Section 2.6 we describe the process of customizing CCASH. In Section 2.7 we outline a case study in which CCASH was used, and in Section 2.8 we discuss conclusions and future work.

2.2 Related Work

Here we present previous work that helped to motivate the feature set outlined in Section 2.1 and to inform the way that CCASH implements those goals. Due to the importance of cost efficiency to those goals, a large portion of the work we cite consists of annotation projects, studies, and tools that were used to develop cost-efficient methods of annotation.

Automatic annotation, or pre-labeling, consists of using NLP algorithms to automatically annotate each instance before it is presented to an expert annotator. Expert annotators then need only review and correct the proposed annotations, which can be much quicker than annotating from scratch. Marcus et al. [29] evaluated automatic annotation using an interface embedded in the GNU Emacs Editor to annotate the Penn Treebank. They manually timed four annotators and found that automatic annotation more than doubled annotation speed and also increased accuracy and inter-annotator agreement. Chiou et al. [9] manually timed two annotators and reported a 70% increase in annotation speed using automatic annotation on a Chinese Treebank annotation task. They did not report the tool they used. Ganchev et al. [18] used a custom web-based tool to do named entity recognition (NER). They evaluated an automatic annotator that presented annotators with a set of plausible guesses instead of a single best guess. They manually recorded the time of a single annotator, reporting a more than 50% increase in speed compared with a manual baseline.

The dramatic time savings reported in these studies underscore the importance of providing proven annotation methods in any general-use annotation framework. Also, notice that each study evaluates automatic annotation by manually timing a very small number of annotators. These results are convincing, but relatively informal. This suggests a need for annotation tools that automatically record cost in such a way as to facilitate exploratory studies, allowing significant user studies to be run without much overhead. Also, flexibility and customization were shown to be important to annotation tools. For example, Ganchev et al. [18] found it necessary to tweak the simple concept of automatic annotation in order to make it successful in the domain of NER.

Many automatic annotators require annotated training data. Annotated data are cheaply available for common tasks in major languages. However, in order to apply automatic annotation to a new task or to a new language, expert annotators must be paid to annotate training data, reducing the cost-efficiency of automatic annotation.

Active Learning (AL) is a technique that addresses this problem by reducing the cost of annotating useful amounts of training data [21, 22, 39, 41]. AL controls which data instances an expert is asked to annotate, presenting them with instances likely to be most informative for learning algorithms. The resulting annotations may be used to train an automatic annotation algorithm.

Ngai and Yarowsky [35] evaluated the effectiveness of AL for noun phrase chunking using an hourly cost model. For this study, seven annotators used a custom-built Java annotation client communicating with a server to enable centralized AL and record timing information. Tomanek et al. [45] evaluated the performance of AL on the task of NER in immunogenetics. They developed and used JANE (the Jena ANnotation Environment), a Java program built on MMAX2 [33], to record annotators' timing information. JANE uses a client-server architecture, allowing distributed annotation and multi-annotator AL.

Both of these studies deal with multiple annotators by centralizing their data and developing tools with client-server architectures. They also extend AL to the multi-annotator case, again underlining the variety of implementations possible for each established annotation method.

Ringger et al. [40] conducted an AL study with 47 annotators doing English POS tagging using a custom-built web application that collected timing information. They used that information to derive an hourly cost model for English POS tagging, which Haertel et al. later incorporated into a cost-conscious version of AL [22]. This is a case where cost measurements were not just used to provide evidence for the effectiveness of a particular method of annotation, but were actually incorporated into an annotation method. In other words, there are some cost-efficient annotation methods that cannot be implemented with an annotation tool that does not record and provide access to cost information, making real-time cost measurement essential.

Representative general-use annotation platforms that influenced CCASH's design include GATE [13], Word-Freak [32], MMAX2 [33], Knowtator [36], and JANE [45]. These tools all support common annotation tasks and also allow for the creation of custom annotation tasks with different degrees of flexibility. GATE is a Java tool that uses a client-server architecture to coordinate multiple annotators, allows timing information to be recorded, and uses a modular design to promote customization. Knowtator allows users to define complex annotation schemas, making it exceptionally configurable and reducing the need for customized plug-ins. MMAX2, like GATE, is a highly modular Java application with a client-server architecture. JANE is a Java application built on MMAX2 that, as mentioned before, provides a form of AL. Word-Freak supports both automatic annotation and searching based on annotation confidence, which allows annotators to engage in a kind of manual AL.

2.3 Web Application Framework

Although the features outlined in Section 2.1 could be implemented in a variety of ways, CCASH designers felt that a web application framework was most fitting for a number of reasons. Previous annotation tools have tended toward client-server relationships in order to centralize data and facilitate multiple annotator collaboration. Web applications make client-server architecture easy and natural. Among the tools described in Section 2.2, GATE and MMAX2 seem to be the most popular due in large part to their support for extensive programmatic customization. A web application seemed a good choice for a customizable architecture, since Internet architecture has a tradition of being extremely customizable, even allowing modules written in different languages and running on different platforms to interoperate.

Being a web application gives CCASH other key advantages in a distributed annotation project. The overhead of configuring a collaborative annotation project can be handled by a single administrator with access to the server. Annotators can then immediately begin annotating texts from any GWT-supported web browser with virtually no per-user configuration time. Since web applications are reloaded every time a user revisits the site or refreshes the browser, there is no difficulty associated with distributing software or project configuration updates. Any updates to the annotation task or to the CCASH framework are instantly and transparently available to all annotators.

2.4 CCASH Architecture

CCASH's architecture consists of four parts: a web client, a web server, a database, and an instance provider (see Figure 2.1).

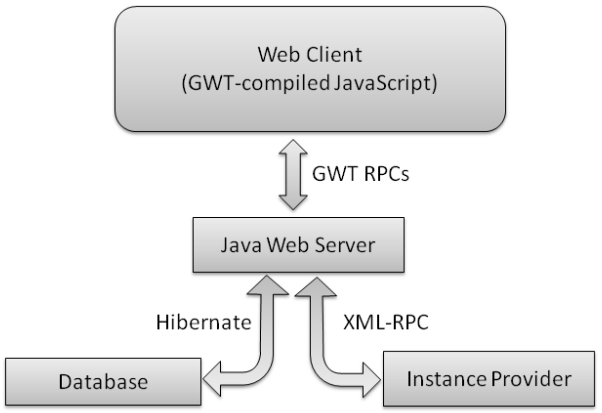


Figure 2.1: CCASH Architecture

2.4.1 Instance Provider

Instance providers are processes with a single responsibility: to provide instances to annotators. In this context an instance is a piece of text, such as a word or sentence, which requires expert annotation. The instances that an instance provider returns may optionally be pre-annotated. Instance providers are largely independent from the rest of CCASH. They make themselves available as web services at some address by implementing a simple XML-RPC interface (see Section 2.5.3). In CCASH, part of setting up an annotation project is giving it the address of a valid instance provider. Because instance providers are decoupled across the network from the rest of CCASH, they may be implemented in any language. This is particularly valuable since instance providers are a prime target for making use of NLP algorithms such as pre-labeling and AL. Algorithm libraries and custom research tools exist in many languages besides Java, and may be reused as part of implementing an instance provider. Because of this network decoupling, instance providers may also be located at anywhere in the world, although because of network latency issues we anticipate that they will commonly be located either on the same machine as the web server, or nearby.

For convenience, CCASH provides Java instance providers that use generics to return any type of instance in sequential and random order. We are also working on including Java instance providers that implement several varieties of pre-labeling and AL.

2.4.2 Data Model

Deciding how to represent and store instances and annotations was a difficult design decision in CCASH. Ideally, one would invent a data structure that is both efficient and also able to encode every instance and annotation type that might be required. For example, a POS tagging task might require annotations to be a sequence of tags. Dependency parsing, on the other hand, might require annotations to contain sets of directed connections between word pairs in the corresponding instance. One can imagine that a data structure able to represent both of these annotations (not to mention a multitude of other possible annotation tasks) would run the risk of being bulky and cumbersome. However, if the data structure were not sufficiently general, it would lose the ability to represent certain tasks and the framework would be unusable for them. Also, if a data structure required users to radically alter their own data schemas in order to fit CCASH's structures, it might discourage them from using the framework.

Recall that one of our high-level design goals is to provide reasonable default implementations whenever possible, always allowing those with unique requirements to define their own functionality. Guided by this principle, we decided to provide some reasonable default data representations and separate the CCASH framework from task-specific data structures as much as possible, allowing developers to use their own data structures, if desired, with minimal interference from the framework.

The two parts of CCASH that need to work with task-specific instance and annotation structures are the web client and the instance provider. The web client must know how to appropriately display the data instances and collect the desired annotations. The instance provider must select instances, possibly pre-label them, and then send them to the client. It receives new annotations from the client, updating its models with new annotations and recording the annotations alongside the data. The web client and the instance provider share a common method of serialization, and between those two endpoints, CCASH is ignorant of instance and annotation values. CCASH simply passes the serialized value along as a member variable of wrapper objects that CCASH uses to maintain records in its own database.

2.4.3 Web Client and Server

The web client is the portion of the application that runs in a user's browser using a combination of HTML and JavaScript. The CCASH client-side framework is written using the Google Web Toolkit (see Section 2.5.1), and we recommend that CCASH developers extending that framework or implementing new annotation tasks (see Section 2.6) do the same. While annotating, the web client's principle responsibility consists of requesting instances from the web server, displaying them to the user, and collecting annotations. The client then sends those annotations back to the web server.

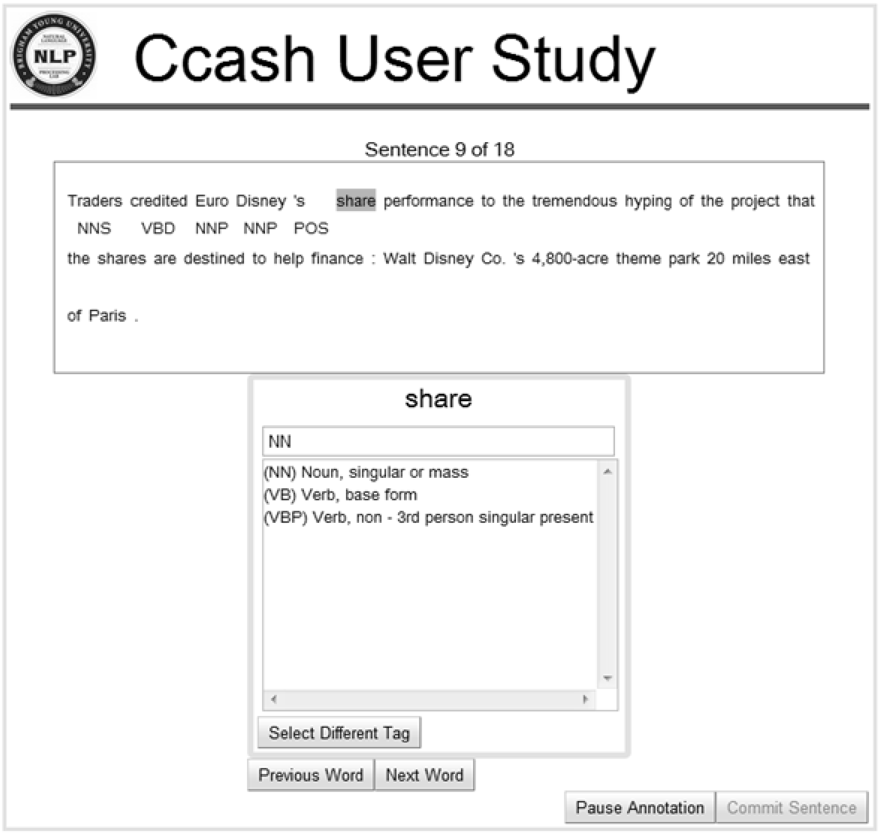
The web server is in charge of facilitating client interactions with other components such as instance providers and the database. It passes on client requests for new instances to the appropriate instance provider and notifies the same instance provider when annotations are completed, giving the instance provider a chance to update its models given this new information.

The CCASH framework uses a combination of client-side interfaces and server-side storage to provide out-of-the-box user account management and project management. It also maintains a database containing information about each annotation (see Section 2.4.5), allowing access to project statistics.

2.4.4 Widget Libraries

In order to make new tasks as easy as possible to implement and customize, we implement default tasks by creating and assembling re-usable GWT widgets. For example, the English POS task in

Figure 2.2 is a combination of a sequential annotation widget (allowing navigation over a sequence of instances), an instance annotation widget (highlighting the current instance in a box) and an English POS instance annotation widget which makes use of an auto-completion widget populated with the Penn Treebank tag set. The auto-completion widget allows users to type in any part of the tag or description, narrowing down selection options to entries that match any part of the selection.



Traders credited Euro Disney 's share performance to the tremendous hyping of the project that
NNS VBD NNP NNP POS
the shares are destined to help finance : Walt Disney Co. 's 4,800-acre theme park 20 miles east
of Paris .

(NN) Noun, singular or mass
(VB) Verb, base form
(VBP) Verb, non - 3rd person singular present

Figure 2.2: English POS Task in CCASH

Because CCASH is intended to be used for research as well as large-scale annotation projects, the framework includes widgets useful for building user studies. These currently include widgets for instructions, surveys, and tutorial annotations with feedback.

In addition to the widgets offered by CCASH, many widgets come standard with GWT, and other third party GWT widget libraries are freely available. Because GWT can interface with native JavaScript, even third-party JavaScript libraries can be used with some additional overhead.

2.4.5 Evaluation

Previous work suggests a strong need for measuring the cost of each annotation in terms of time [21]. This is not, however, the only possible measure of cost. Culotta and McCallum [12], for example, measure cost in terms of the number of required user actions to fix an annotation in a given user interface. This is a reasonable surrogate for time, since more interactions generally mean more time, and it enjoys the benefit of being easy to predict. CCASH provides a flexible mechanism for measuring cost by collecting events fired by the web client into a simple sequence analogous to a timeline. Each timeline event has a name and a timestamp, allowing calculation of cumulative time, number of interactions, and other desired statistics. CCASH by default fires events when an annotation instance is requested, when it is presented to an annotator, when an annotation is completed, and when an annotation task is paused or resumed. If more granularity is required, for example if each user interaction needs to be recorded, CCASH developers implementing new tasks in CCASH can fire custom events at any point.

This cost information can be used to evaluate cost-reduction strategies post hoc. But it can also be used by an annotation method that learns from annotation costs. Haertel et al. [22] and Settles et al. [42] have both proposed methods of incorporating cost models into the AL process, helping to offset traditional AL's bias towards long, costly instances.

2.5 Core Technologies

CCASH makes use of several supporting technologies. This section briefly describes what they are and how they are used.

2.5.1 Google Web Toolkit (GWT)

CCASH's web client component is implemented with the Google Web Toolkit (GWT). GWT allows developers to build user interfaces in Java using familiar Swing-like widgets. GWT provides a cross-compiler that compiles Java code into optimized JavaScript which communicates with a Java web server using remote procedure calls. GWT packages this entire bundle—JavaScript for the client and Java code for the server—into a Web Archive (WAR) which can be hosted on any compatible Java web server like Apache's Tomcat or Red Hat's jBoss.

We chose to use GWT to implement the web client portion of CCASH for several reasons. Most importantly, GWT helps user interface developers abstract away from the browser-specific idiosyncrasies that can make web programming difficult for newcomers. CCASH is designed with the assumption that future researchers who create new tasks for CCASH will likely be familiar with Java programming and at least one of the two major interface design paradigms that GWT supports: assembling Swing-like graphical components programmatically, or else defining XML interfaces bound to Java objects (similar to more traditional web-page design). GWT code compiles to JavaScript that is compatible with most major modern web browsers including IE, Firefox, Safari, and Opera. GWT also provides several mechanisms for creating localizable web applications. This helps CCASH support Unicode and right-to-left languages as well as locale-specific text and styles. Also, GWT facilitates using the browser history buttons to navigate through locations within a web application by encoding some application state in a history token embedded in the browser's address bar.

2.5.2 Hibernate

The Java Persistence API is a robust and standard way to manage permanent data storage in Java. We chose to use Hibernate to implement this API and to interface with the database layer of CCASH. This means that if developers find that they need to persist their own custom data objects, they can do so by simply annotating their data object classes in compliance with the Java

Persistence API. It also means that framework users are free to use any of the many database implementations that are supported by Hibernate.

Using the Java Persistence API makes it easy to place the database either on the same machine that is running the web server or on any other machine that is network-accessible. Note that storing annotations in a database makes them efficiently accessible without precluding the possibility of exporting them to other formats such as XML.

2.5.3 XML-RPC

XML-RPC is a simple protocol for making remote procedure calls over the network. Because of the simplicity of the protocol, implementations exist in many programming languages including C, C++, C#, Java, Python, Ruby, Lisp, and many more. This makes it easy for the instance provider to be implemented in almost any language in order to reuse existing algorithms implementations or libraries for automatic annotation and AL.

2.6 Defining Custom Tasks

The process of adding a new annotation task to CCASH consists principally of creating a client-side user interface for the task and then connecting it to an appropriate instance provider. The following subsections describe this process in more detail.

2.6.1 Building a Client-side User Interface

In the CCASH framework we have implemented an English part-of-speech (POS) annotation task (Figure 2.2) and a named entity recognition (NER) annotation task (Figure 2.3). In both of these tasks, the user is presented with an interface that gives context at the top of the page and a more focused inspector that we call the "lens" below. When implementing a new task in the web client, developers may either take advantage of this pre-existing layout or else build their own layout.

To build custom client-side interfaces, developers extend a helper class that takes care of bookkeeping such as firing standard timeline events (see Section 2.4.5). They then create and

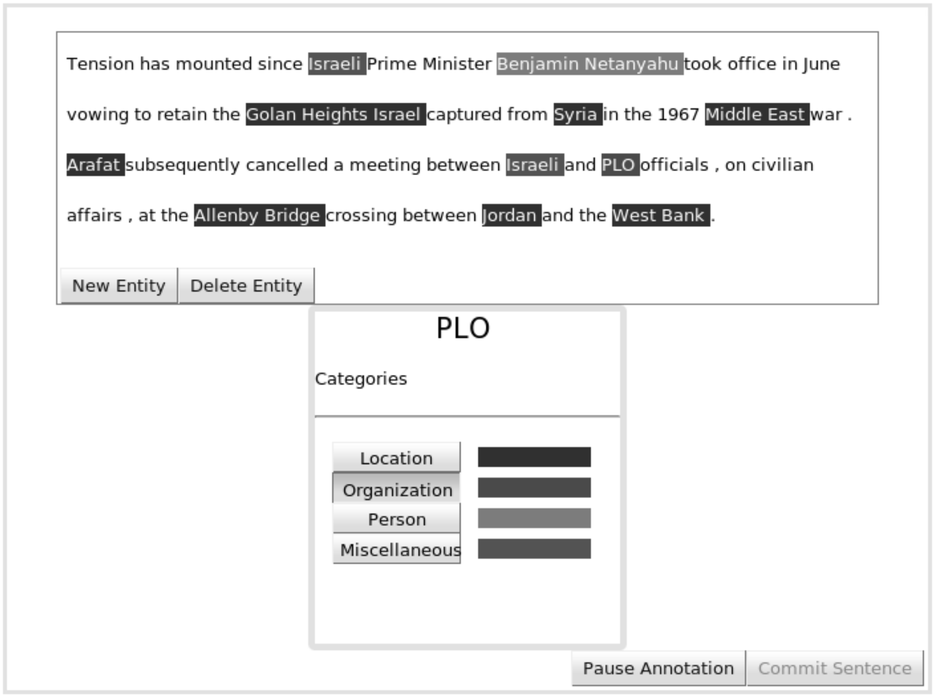

Tension has mounted since Israeli Prime Minister Benjamin Netanyahu took office in June vowing to retain the Golan Heights Israel captured from Syria in the 1967 Middle East war . Arafat subsequently cancelled a meeting between Israeli and PLO officials , on civilian affairs , at the Allenby Bridge crossing between Jordan and the West Bank .

Figure 2.3: Named Entity Recognition (NER) Task in CCASH

assemble the widgets necessary to implement their task. As mentioned before, third party widget libraries are available for GWT. CCASH also provides widgets for handling common high-level tasks such as navigating within a sequence of instances and highlighting the instance currently in focus. If no helper widgets fit a given task, a developer is free to implement that task from scratch.

Finally, a task designer who is interested in task-specific timing information will want throw custom timeline events in the web client at the appropriate times.

2.6.2 Building an Instance Provider

Creating an instance provider consists of implementing an XML-RPC interface whose most important method allows clients to get the next instance for a particular annotator. As explained in Section 2.5.3, instance providers need not be implemented in Java. However, if a new instance provider is implemented in Java, it can make use of convenience methods for filtering timing events, serialization, and XML-RPC implementation.

CCASH includes Java helper classes with generic instance and annotation types that may be used to implement instance providers with a variety of data types. These helper classes currently support only trivial instance orderings (sequential and random), but we hope to soon provide a complete AL and pre-labeling framework.

2.7 Case Study

One of CCASH's principle objectives is to facilitate exploratory studies and comparisons of different annotation methods. CCASH has been used for that purpose in a recent user study conducted by Carmen et al. [6]. In this study CCASH was used to record the times of a group of thirty-three linguistic graduate students as they annotated Penn Treebank sentences with English POS tags. They were given additional help in the form of suggestions from a POS tag dictionaries, which consist of simple mappings from each word type to tags that were previously applied to that type. The coverage level of such dictionaries was shown to have an impact on annotation time and accuracy.

Carmen et al. had some non-trivial constraints on study organization. The study presented each participant with a common set of 18 sentences in the same order with one of six different POS tag dictionary coverage levels. Additionally, the study ensured that each user encountered each coverage level exactly three times, and also that each coverage level was applied to a sentence of significantly different length.

These constraints affected both the order in which sentences were provided to different users and the quality of suggestions offered to the participants. Because of this, we feel that this study provides some evidence for CCASH's ability to handle diverse annotation methods in practice.

Additionally, after setting up CCASH for the study, very little effort was required to run it to completion. Subjects worked from a variety of locations using a variety of web browsers. Administrators were able to monitor the progress of the study from the administrator interface, downloading and reviewing statistics periodically. When one user encountered a minor bug, it was fixed without requiring the participants to reinstall or update any software. Also, data and annotations were collected centrally, eliminating any need to distribute data or collect resulting annotation or timing information.

2.8 Conclusions and Future Work

CCASH is a web application framework designed to give researchers and corpora builders a common platform for developing cost-efficient annotation methods and for applying them in annotation projects. CCASH currently shows promise in meeting these goals by supporting two common tasks: POS tagging and NER labeling. It has also been successfully used as a platform for a user study evaluating the effectiveness of using POS tag dictionaries to speed up English POS tagging.

We are making the entire CCASH project public on SourceForge.net (`http://sourceforge.net/projects/ccash`). As we improve the process of extending CCASH with new annotation tasks, we hope that the language resources community will begin to contribute their own annotation tasks, share useful widgets, and collaborate on the framework. At the same time we plan to release a Java framework for AL and automatic annotation.

Additionally we plan to extend CCASH to implement the OpenID protocol (`http://www.openid.net`) so that users can log in with any OpenID provider, avoiding the annoyance of creating a dedicated CCASH account.

Finally, we are currently implementing a Syriac morphological annotation task in CCASH. Because Syriac is a low-resource language and Syriac morphological annotation is a non-trivial task, expert annotators are expensive. It will be important to quickly determine which annotation methods are most cost-effective, and CCASH will be a good means to accomplish this. This Syriac annotation task will involve a number of annotators dispersed around the world. We are interested in experimenting with different cost-conscious methods for coordinating the efforts of multiple annotators and in building successful approaches into the CCASH framework.

2.9 Acknowledgements

We would like to thank Jeremy Sandberg for his contributions to the code base for this project.

Chapter 3

First Results in a Study Evaluating Pre-annotation and Correction Propagation for Machine-Assisted Syriac Morphological Analysis

Author List

Paul Felt, Eric Ringger, Kevin Seppi, Kristian Heal, Robbie Haertel, Deryle Lonsdale

Publication Venue

The Eighth International Conference on Language Resources and Evaluation (LREC) 2012

MS Thesis Context

This paper fits into the larger thesis by describing a user study gathering information needed to answer the thesis question. It also conducts a preliminary analysis of the data that indicates that high accuracy pre-labels increase annotator accuracy and speed.

Abstract

Manual annotation of large textual corpora can be cost-prohibitive, especially for rare and under-resourced languages. One potential solution is *pre-annotation*: asking human annotators to correct sentences that have already been annotated, usually by a machine. Another potential solution is *correction propagation*: using annotator corrections to dynamically improve to the remaining pre-annotations within the current sentence. The research presented in this paper employs a controlled user study to discover under what conditions these two machine-assisted annotation techniques

are effective in increasing annotator speed and accuracy and thereby reducing the cost for the task of morphologically annotating texts written in classical Syriac. A preliminary analysis of the data indicates that pre-annotations improve annotator accuracy when they are at least 60% accurate, and annotator speed when they are at least 80% accurate. This research constitutes the first systematic evaluation of pre-annotation and correction propagation together in a controlled user study.

3.1 Introduction

The current success and widespread use of data-driven techniques for processing human language make annotated corpora an essential language resource. For instance, many popular natural language processing (NLP) algorithms require significant amounts of high quality annotated training data in order to perform effectively. Also, annotated text can be useful in its own right as a means of exploring the language and the culture that produced it. For example, one might use syntactic annotations to study discourse patterns, or topical annotations to track the movement of important ideas through time and space.

Scholars at the Center for the Preservation of Ancient Religious Texts (CPART) of the Neal A. Maxwell Institute for Religious Scholarship at BYU and at the Oriental Institute at the University of Oxford are jointly working on a project called the Syriac Electronic Corpus, with the goal of creating a comprehensive, labeled corpus of classical Syriac. Classical Syriac ('kthobonoyo') is an under-resourced Semitic language of the Christian Near East and a dialect of Aramaic. It was largely replaced by Arabic as a spoken language by the end of the ninth century, and is now primarily a liturgical language. Many prolific authors wrote in Syriac. The goal of the Syriac Electronic Corpus project is to annotate all of these texts with morphological information to facilitate systematic study of Syriac by historians, linguists, and language learners.

ܠܡܰܠܟ݁ܳܟ݂ܽܘܢ	ܡܰܠܟ݁	ܡܠܟܐ	ܡܠܟ
token	stem	citation form	root

Figure 3.1: The Syriac word token LMaLK'K,uON "to your king" and its related forms

Morphological analysis of Syriac involves segmenting a word into its constituent morphemes and labeling each according to its grammatical form(s). For our purposes, a word token consists of a prefix, a suffix, and a *stem*, which we define as the remaining text. The dictionary citation form (or baseform) and, where applicable, the root are identified from the stem (Figure 3.1).

In contrast to English, where searching for a few forms of a word is often sufficient for discovering patterns reflecting the word's usage and meaning, in Semitic languages search and discovery are not so straightforward. If we could search Syriac texts on citation forms or even on roots, we could search for and discover patterns as easily as in English; however, Syriac roots are altered by extensive inflectional and derivational morphological processes such that numerous surface forms correspond to any given root. As a result, searching Syriac text is ineffective since one must either limit one's query to a single inflected surface form or use heuristics to expand the query, buying higher recall at the price of lower precision.

A morphologically annotated digital corpus of a lesser studied language lends itself to search and therefore to careful study in a way that formerly only experts could attempt based on long years of familiarity. Such annotated corpora enable scholars to study and discover the contributions of and trends in historical documents. One outstanding example of such a corpus is the Dead Sea Scrolls Electronic Library, assembled by CPART scholars [46]. The Syriac Corpus will be an artifact of similar value to linguists, Syriac students, and scholars of Syriac, the Near East, and Eastern Christianity.

Unfortunately, creating annotated corpora can be extremely time-consuming. The Way International Foundation, a Biblical research, teaching, and fellowship ministry, spent 15 years labeling the Syriac New Testament with morphological annotations [25]. The Syriac New Testament consists of approximately 100,000 words. Similarly, two Syriac scholars we worked with during the course of this research informally report taking two years to label about one fourth of the Old Testament. By contrast, the Syriac Corpus will encompass over 10,000,000 words. To achieve this goal in a timely manner it will be necessary to increase the speed of annotation.

Pre-annotation, also known as pre-labeling, has the potential to reduce annotation cost by using NLP algorithms to automatically annotate each instance (i.e. sentence) before it is presented to an expert annotator. Expert annotators then need only review and correct the proposed annotations, which can potentially be done much more quickly than annotating from scratch.

Kristjansson et al. [27] describe an enhancement to pre-annotation for multi-part annotation tasks which they call correction propagation. Correction propagation consists of triggering a pre-annotation update whenever an annotator corrects a pre-annotation. The idea is that the machine annotator can use the correction to improve its guesses regarding other decisions to be made for the item currently being annotated (e.g. sentence). Kristjansson et al. give the example of identifying contact information in free text. In this case, correcting a pre-annotated given name might allow the automatic annotator to correctly identify a corresponding surname and address. To be clear, correction propagation does not involve retraining a model using the new data. Rather, it involves making a multi-part prediction in a hypothesis space that is constrained by a partial annotation.

Both pre-annotation and correction propagation require a model capable of supplying automatic annotations, and correction propagation additionally requires the ability to constrain and update automatic annotations. However, as noted earlier, many NLP algorithms for building such a model require previously annotated training data. For tasks and languages without already existing resources, one must therefore begin the annotation process with low quality pre-annotations and periodically retrain the pre-annotator as more data is labeled. Although pre-annotation and correction propagation attempt to increase annotator efficiency, it is conceivable that inaccurate predictions could reduce annotator speed or accuracy. Because of this, before building annotated corpora in domains with little labeled data, it is desirable to have a sense of how accurate a model must be in order to make pre-annotation and correction propagation helpful instead of harmful. This research constitutes the first systematic evaluation of pre-annotation and correction propagation together in a controlled user study.

3.2 Related Work

In order to generate pre-annotations and correction propagation updates for Syriac morphological analysis, we use Syromorph, a probabilistic morphological analyzer for Syriac described by McClanahan et al. [30]. Syromorph is an n-best pipeline of classification and transduction tasks. Each task in the pipeline proposes hypotheses based on the data and the results of all previous tasks.

Solutions are chosen by running a beam search over all the hypotheses in the pipeline, allowing decisions to be made in a global context without incurring the cost of full joint inference. Syromorph first segments each word into its parts: prefix, stem, and suffix. Syromorph then predicts a baseform, or dictionary citation form, for the stem. Finally, Syromorph predicts the grammatical attributes of the stem and suffix. [1]

Pre-annotation has been evaluated on a variety of tasks. Marcus et al. [29] evaluated pre-annotation using an interface embedded in the GNU Emacs Editor to label the Penn Treebank with English part-of-speech (POS) tags. They manually timed four annotators and reported that pre-annotation more than doubled annotation speed and also increased accuracy and inter-annotator agreement. Chiou et al. [9] timed two annotators using an unspecied tool and reported a 70% increase in annotation speed using pre-annotation on a Chinese Treebank annotation task. Baldridge and Osborne [1] present several choices rather than the single best for a parsing task and report a 74% reduction in cost. Similarly, Ganchev et al. [18] present a set of candidate pre-annotations to annotators doing named entity recognition. They manually recorded the time of a single annotator and reported a more than 50% increase in speed. Brants and Plaehn [5] applied pre-annotation to parse tree labeling. In order to make pre-annotation effective for parse tree labeling, they found they had to alter their pre-annotation approach by creating an interactive parse tree where annotators accept or reject suggestions starting at the parse tree's leaves and working their way to the root. These results are encouraging, but it is unclear which, if any, of the previous pre-annotation results

[1] In accordance with the current needs of the Syriac Corpus project, the original Syromorph (v1.0) has been modified slightly so that it no longer predicts a root form (current version is 2.1). The reason for this change is that the ultimate goal of the project is to link each token to a baseform dictionary entry, and the root form comes for free with this linkage.

apply to Syriac morphological analysis or to other linguistic annotation tasks. For one thing, pre-annotation had to be adapted before it was effective for some of the tasks. Most importantly, all previous work evaluates only the best available quality pre-annotations.

Correction propagation has been evaluated on far fewer tasks than pre-annotation. As has already been noted, Kristjansson et al. [27] applied correction propagation to the task of information extraction, interactively assisting users to fill in database fields. They evaluated the performance of correction propagation in simulation and showed that automatic annotator accuracy significantly increased after even a single correction. They also showed that correction propagation significantly reduced the expected number of user interactions with a hypothetical graphical user interface. These results are promising; however, because of the differences between information extraction and Syriac morphological analysis, it is unclear how helpful correction propagation will be for Syriac morphological analysis.

3.3 Methodology

This section describes the conditions under which the data was collected; a preliminary analysis of the data is described in Section 3.4.

This section will proceed as follows: sub-section 3.3.1 gives an overview of the user study layout; 3.3.2 describes the training and evaluation of the automatic annotation models used in the study; 3.3.3 shows via simulation that correction propagation has the potential to increase effective pre-annotation accuracy; 3.3.4 explains our method of assigning experimental conditions to participants; 3.3.5 describes the user study participants; 3.3.6 describes the framework used to conduct the study and the study's graphical user interface.

3.3.1 User Study Overview

We designed a web-mediated user study using CCASH,[2] an open source web application framework for linguistic annotation tasks [15]. In the study, annotators took a survey, received a brief

[2] `http://ccash.sourceforge.net`

training, and then worked through four practice sentences. After each practice sentence, participants received feedback on how their annotations differed from the annotation guidelines they were given. They were required to achieve a high level of accuracy on the final practice sentence before proceeding. Finally, participants annotated 30 sentences under a sequence of randomly assigned experimental conditions, explained in Section 3.3.4. For each word in the study, CCASH recorded the time each annotator took to spent as well as the number of correct and incorrect decisions they made.

The choice to have all participants annotate the same 30 sentences does not limit our ability to collect large amounts of data and identify statistical trends associated with different annotation conditions. It does limit the applicability of our results to new data; however, that is a problem inherent in any focused study.

A gold standard annotation was constructed by two expert Syriac linguists who completed the study, then discussed and resolved all disagreements in their annotations. It should be noted that annotated Syriac text already exists: The Syriac Peshitta New Testament has been labeled with morphological information [25]. However, reference copies of this data have been published which could bias the results of our study. Accordingly, the 30 sentences for the study were selected uniformly at random from The Acts of Judas Thomas, an apocryphal text that is similar, but not identical, to the New Testament [48].

When constructing a gold standard, it is important to acknowledge that there are some difficult cases that even experts have difficulty agreeing on [3]. However, the disagreements between our experts indicated that only around 20 of the 1289 decisions in the user study were difficult. This rate is low enough that it should not greatly affect our results.

3.3.2 Model Training and Metrics

We trained Syromorph models on various random subsets of the Syriac New Testament data assembled by Kiraz [25] and augmented with suffix data by McClanahan et al. [30], consisting of approximately 100,000 labeled tokens. We calculated model accuracy against the 30 Judas Thomas sentences in the study's gold standard. This slight mismatch between model training and test data caused model accuracy to suffer. Thus our most accurate model, trained on all of the New Testament data, achieved an accuracy of only slightly above 90%. In order to obtain models with given target accuracies, we trained Syromorph on random subsets of the training data until a model was found which achieved the desired accuracy ±0.01% measured against the gold standard.

In a multi-part annotation task like Syriac morphological analysis, accuracy can be calculated on the sentence level, the word level, or the decision level. These accuracy metrics are highly correlated, but not identical. Furthermore, since decisions can be partitioned into classes according to their sub-task, it is possible to calculate decision- level accuracy either as a macro-average or as a micro-average across decision types. A macro-average is computed by first averaging the decisions for a sub-task, then averaging the resulting averages. A micro-average is computed by averaging the decisions for all sub-tasks at once. Decision-level accuracy using a micro-average is an appropriate accuracy metric since it is computed over the exact set of choices that an annotator must make while annotating. All accuracies mentioned in this paper are decision-level micro-averages calculated against the 30 sentence gold standard set.

3.3.3 Simulated Correction Propagation

Before conducting a user study to test whether correction propagation reduces annotation effort in a scenario involving real users, we ran simulations to verify that correction propagation has the potential to increase effective pre-annotation accuracy.

In the first series of simulations, referred to in Figure 3.2 as "Without Correction Propagation," Syromorph models trained on increasing amounts of data were queried for labels a sentence at a time. In the second series of simulations, referred to in the figure as "With Correction Propagation," the same models were queried for labels a decision at a time, constrained by a correct partial labeling of all previous decisions in the sentence. This measures the accuracy of the pre-annotations an infallible annotator would encounter working sequentially through the decisions of

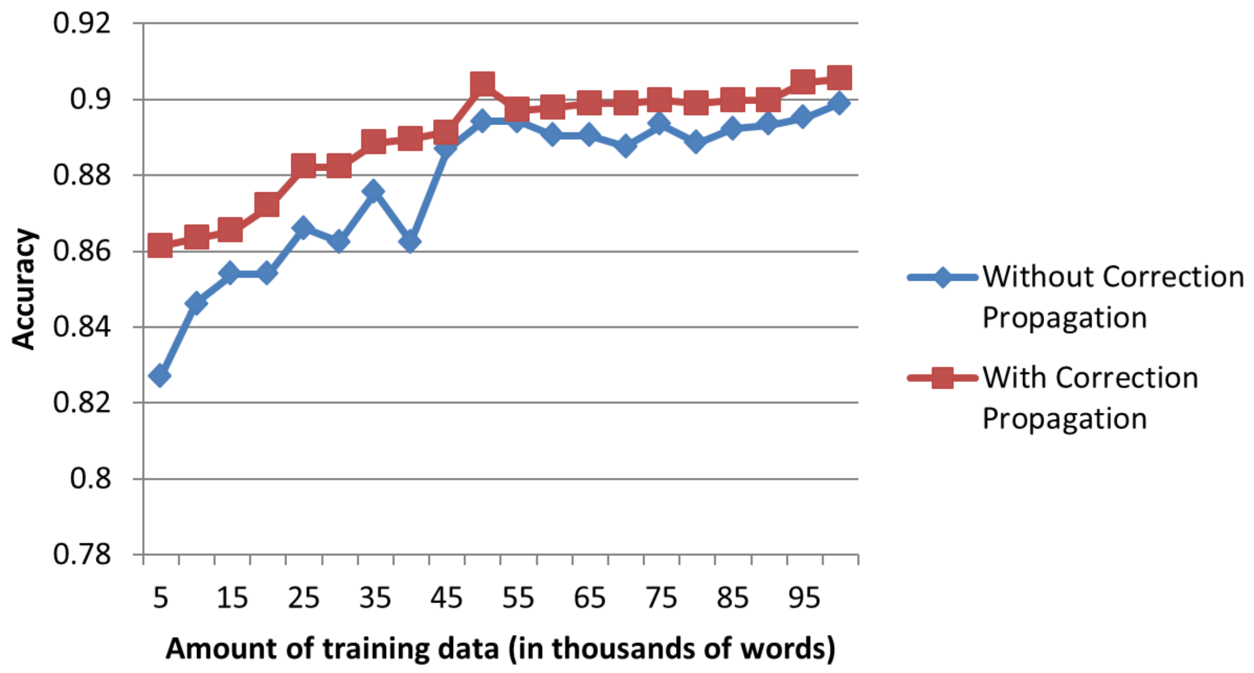


Figure 3.2: Syromorph's accuracy with and without correction propagation.

each sentence, where the model was allowed to update the sentence's pre-annotations after each decision.

Figure 3.2 shows that correction propagation allows models at all quality levels to improve the accuracy of their decisions by a modest amount. These simulations indicate that correction propagation has the potential to increase pre-annotation accuracy in practice. This increased pre-annotation accuracy could also conceivably increase annotator speed, since a more accurate pre-annotation will usually be easier to correct.

3.3.4 Experimental Conditions

Pre-annotations were supplied to annotators at the following accuracy levels: none, 25%, 35%, 45%, 55%, 65%, 75%, 90%, and 100%. In the none case, no pre-annotations were given. In the 100% case, gold standard annotations were given. In all intermediate cases, Syromorph models trained to the indicated accuracy provided pre-annotations. The accuracy levels between 25% and

	Prt1	Prt2	Prt3	Prt4	...	Prt16
St1	0	25	25+CP	36	...	100
St2	25	25+CP	36	36+CP	...	0
St3	25+CP	36	36+CP	47	...	25
...	...	...	...	...	...	...
St16	100	0	25	25+CP	...	90+CP

Figure 3.3: Experimental condition assignment scheme.

90% inclusive were chosen to span the range of accuracies achievable by Syromorph trained on the Peshitta New Testament.

Additionally, participants annotated sentences both with and without the assistance of correction propagation. Note that correction propagation requires a model; consequently it cannot be applied to the none or 100% cases. In all, there are

$$|\{none, 100\}| + |\{25, 36, 47, 58, 68, 79, 90\} \times \{+CP, -CP\}|$$

or 16 parameter combinations to test. We refer to each parameter combination as an *experimental condition*.

It is convenient to assign experimental conditions to participants and sentences using the matrix in Figure 3.3 where *Prt1* is the first participant to take the study, *St1* is the first sentence in the study, and cell values indicate a pre-annotation quality (25-100) and the optional presence of correction propagation (+C).

This matrix can be duplicated indefinitely to the right and the bottom. That is, Annotator 17 can be assigned to the same column as Annotator 1, and Sentence 17 can be assigned to the same row as Sentence 1. This parameter assignment scheme has some nice properties. It guarantees that each annotator encounter each experimental condition roughly the same number of times. It also ensures that each sentence will be encountered under each condition roughly the same number of times. However, this parameter assignment scheme has an important flaw: annotators encounter sentences of steadily increasing quality. Such an apparent trend may affect the way that annotators interact with the pre-annotations. This problem is resolved without sacrificing the nice properties of the assignment matrix by first permuting the rows of the matrix and afterwards the columns. Annotators thus encountered the study's sentences in a fixed order and under every experimental condition, but without an easily discernible pattern.

It may be expected that annotators will begin to annotate slowly then move more quickly as they grow used to the task; this could potentially have a confounding effect on our timing data. We dealt with this learning effect in two ways. First, the training and practice at the beginning of the study allowed participants to become accustomed to the task and interface. Second, the parameter assignment scheme ensured that the sentences annotated under a given experimental condition include approximately equal numbers of sentences annotated early and late in the annotation process.

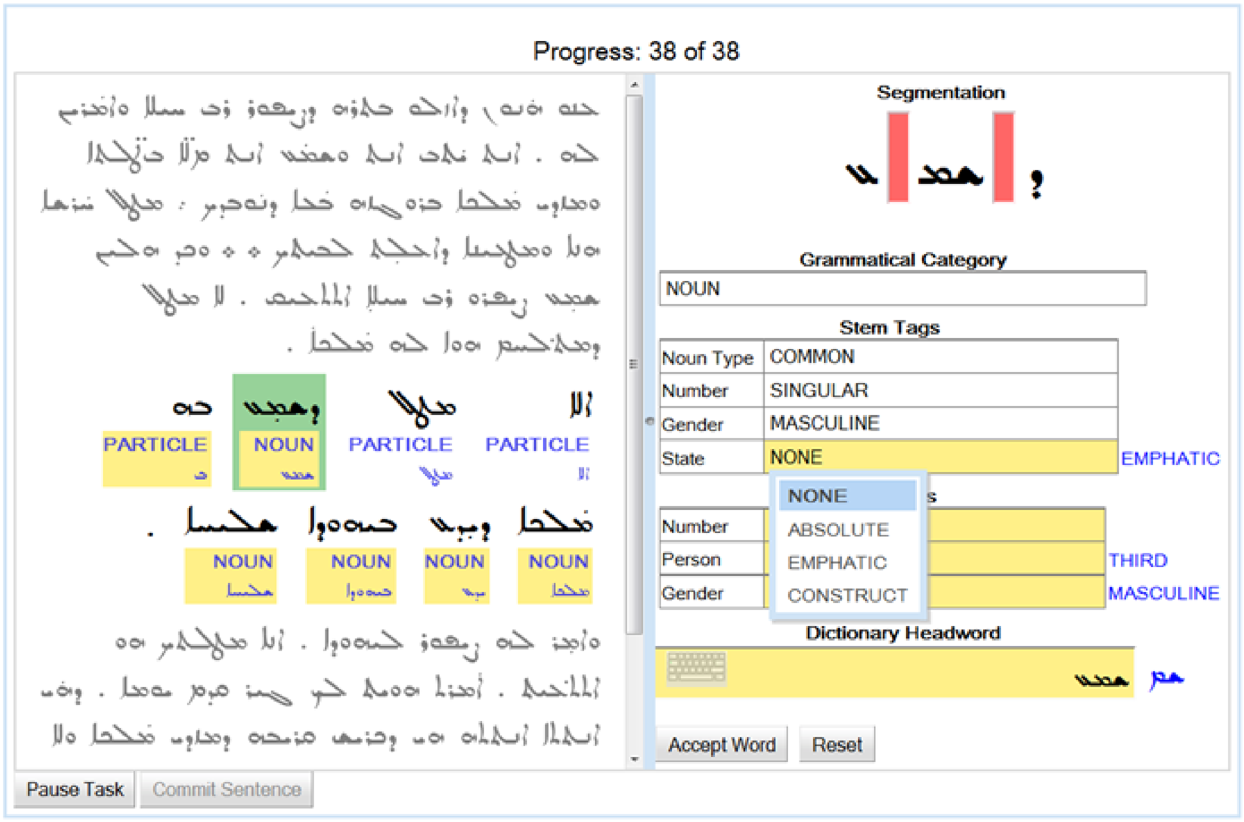
3.3.5 User Study Participants

Nine Syriac experts, invited by colleagues associated with CPART and the Oriental Institute at the University of Oxford, successfully completed the study. Their answers to the survey at the beginning of the study indicated that all participants consider themselves reasonably proficient in Syriac and comfortable using of computers.

3.3.6 Graphical User Interface

The graphical user interface used to conduct Syriac morphological analysis, implemented in CCASH, is an important part of this study since it affects annotation speed and also the applicability of this study to other tasks. Some time was spent refining the interface with Syriac experts to make sure it is reasonably efficient.

Annotators work through a sentence at a time. The sentence being annotated, along with some text preceding and following, is shown on the left side of the screen (see Figure 3.4A). Annotators navigate from one word to another in the sentence either by using clicking on the desired word, or by holding down control on the keyboard and navigating with the arrow keys. Within each word, annotators begin by segmenting prefixes and suffixes using either mouse clicks or a



Grammatical Category: NOUN

Stem Tags

Noun Type	COMMON	
Number	SINGULAR	
Gender	MASCULINE	
State	NONE	EMPHATIC

Number		
Person		THIRD
Gender		MASCULINE

Figure 3.4: The graphical user interface for Syriac morphological analysis used in the study.

keyboard shortcut in Figure 3.4B. Then a grammatical category is chosen in Figure 3.4C (in the example, NOUN), after which a set of stem and suffix tags appear in Figure 3.4D that are applicable for the chosen segmentation and grammatical category. Annotators set tag values either by clicking on them with a mouse and selecting a value from the resulting drop-down list, or else by typing them using a keyboard. For annotators who choose to type, the text is autocompleted for them based on the values that are applicable to that field. Finally, annotators may input Syriac text either by using their mouse to click keys on a virtual keyboard, or by using their keyboard directly in Figure 3.4E.

Once an annotator changes a field value, that field's background changes color. When correction propagation is active, each time the annotator changes a field, the model is queried for a new prediction constrained by all of the decisions that the annotator has made so far in the sentence.

In the scope of the word currently being annotated, if the new pre-annotation differs from the old preannotation, the new value is displayed as a hyperlink to the right of its target field as shown in Figure 3.4F. For all other words in the sentence, pre-annotation values are simply updated in place.

As annotators proceed, CCASH records detailed information about each word including accuracy, the time each element spent in focus, mouse clicks, and the number of keystrokes. To ensure that timing information is accurate, participants are instructed to press the pause button on the bottom left of Figure 3.4 whenever they take a break. Whenever the task is paused, the screen is also obscured.

3.4 Preliminary Analysis

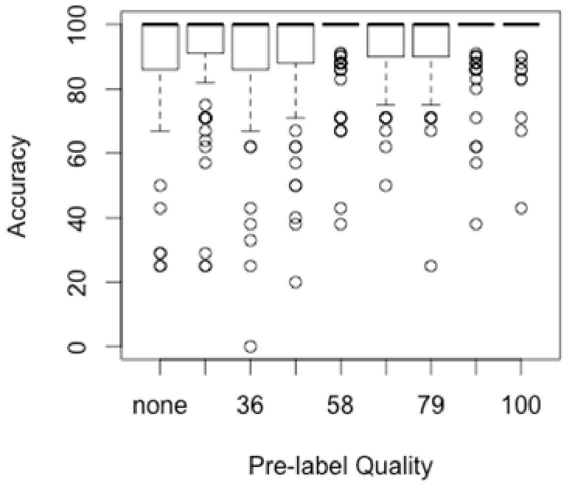
Annotations produced under the same experimental conditions are treated as samples and used to test the various hypotheses of the experiment. In this section, we describe the data and its analysis in more detail.

3.4.1 The Data

Although participants labeled a sentence at a time, it is problematic to do time analysis on the sentence level because the length of each sentence clearly affects its cost, making annotation time difficult to compare across sentences. Controlling sentence length could alleviate this problem, but introduces a new problem since the length-controlled sentences are not representative of the data as a whole. We avoid these difficulties by doing analysis on the word level.

To estimate word annotation times, we record the time that each word was in focus in the GUI. This time is not a perfect stand-in for the time an annotator spent actually working on each word, since it is possible for an annotator to consider a word that is not actually selected. Also, the first word of each sentence will naturally tend to be selected longer than other words in the sentence as an annotator orients herself by reading the sentence and context. However, given sufficient data, these times should be an acceptable approximation for the true time spent annotating each word.

(a) Accuracy per word



(b) Time per word

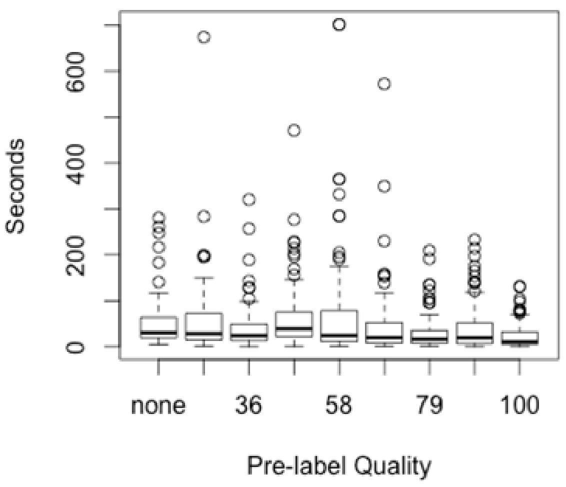


Figure 3.5: Box plots representing the data collected so far at each level of pre-annotation. Data generated using correction propagatation are not included here.

We compute word annotation accuracies by calculating the accuracy of the decisions applicable to the word, as explained in Section 3.3.2.

The study's 9 participants each annotated 30 sentences, or 152 words, resulting in 1,368 word-level data points both for annotation time and accuracy. Since there are 16 experimental conditions, each condition has roughly 85 data points. Figure 3.5 uses standard box plots to summarize the data collected under each pre-annotation condition. Corresponding plots for correction propagation are not shown due to space constraints. Notice that for each condition there is considerable variance in both the accuracy of words annotated (3.5a) and the time required to annotate each word (3.5b).

3.4.2 Hypothesis Tests

Our goal is to use data gathered in the study to determine when pre-annotation and correction propagation improve accuracy and increase speed. A simple way of doing this is by comparing the means of various groups of data and testing whether they are significantly different using null hypotheses. We pose three pairs of null hypotheses.

The first pair of null hypotheses is that annotator speed and accuracy are not significantly different for words annotated with and without pre-annotations. Testing these hypotheses at each of the eight pre-annotation accuracy levels indicates when the pre-annotation ought to be used.

The second pair of null hypotheses is that annotator speed and accuracy are not significantly different for words annotated without assistance and those annotated with the *combination* of pre-annotation and correction propagation. Testing this hypothesis at each pre-annotation accuracy level indicates when combined pre-annotation and correction propagation ought to be used.

The third pair of null hypotheses attempts to tease apart the effects of correction propagation and pre- annotation: assuming pre-annotations are being used, annotator speed and accuracy are not significantly different for words annotated with and without correction propagation. Testing this hypothesis at each pre-annotation accuracy level indicates when correction propagation ought to be used above and beyond pre-annotation.

Each null hypothesis is tested using both a standard two-sided Student's *t*-test as well as a permutation test [31]. The Student's *t*-test is used since it is widely understood and used. A two-sided *t*-test is appropriate since there is the possibility that accuracy and annotation time will either increase or decrease. The permutation test is used since it does not rely on assumptions about any underlying distribution. Note that with 48 null hypotheses being tested, we expect a few spurious rejections. This can be seen by recalling that if we draw two sets of data from the same process, we expect a standard *t*-test with a p-value threshold of 0.05 to incorrectly reject the null hypothesis one time in twenty. However, if pre-annotation and correction propagation do indeed improve annotator time or accuracy, there should be clear trends in the rejections.

3.4.3 Results

Table 3.1 shows the difference between the mean annotator accuracies (a) and times (b) of words annotated under the control condition and of words annotated under the test condition at various

(a) Change in Mean Word Accuracy

Control	**Test**	**Pre-annotation model quality (versus gold standard)**							
Condition	**Condition**	25%	36%	47%	58%	68%	79%	90%	100%
“none”	PA	+2.6	+0.3	+2.5	**+5.4**	**+4.8**	+4.6	**+5.8**	**+7.8**
“none”	PA+CP	+3.1	+2.8	+1.9	+1.9	+3.8	+4.7	**+5.4**	NA
PA	PA+CP	+0.5	+2.5	-0.6	-3.5	-1.0	+0.1	-0.4	NA

(b) Change in Mean Word Time (sec)

Control	**Test**	**Pre-annotation model quality (versus gold standard)**							
Condition	**Condition**	25%	36%	47%	58%	68%	79%	90%	100%
“none”	PA	+5.4	-9.9	+11.1	+15.4	-7.1	**-20.0**	-10.4	**-27.6**
“none”	PA+CP	-7.0	+5.0	+3.1	-8.0	-11.2	-3.9	-2.9	NA
PA	PA+CP	-12.5	+14.9	-8.0	-23.4	-4.1	+16.1	-7.5	NA

Table 3.1: The difference in the mean accuracy (a) and time (b) of words annotated under two experimental conditions: pre-annotation (PA) and correction propagation (CP). Statistical significance at or below the 0.05 level is indicated by underlining for the two-sided *t*-test and bolding for the permutation test.

levels of pre-annotation quality. Increases in accuracy are good and decreases in time are good. Removing outliers has little effect on the outcomes, so we leave them in for all analyses.

In the first row of Table 3.1a, which compares the accuracy of words annotated without pre-annotations to those annotated with pre-annotations, there is a clear block of significant results. It appears that pre-annotations generated by models of quality 60% or higher increase average annotator accuracy by 5-7%, and that increase is usually greater than can be explained by the natural variance of the data. This is an encouraging result for those contemplating using pre-annotation on similar tasks. Although 60% appears relatively high in the range of model accuracies that we have presented, it is actually quite low for a reasonable predictive model. That is, 60% accurate models can be attained with relatively little data for most tasks (in our case roughly 50 annotated sentences), resulting in a low barrier to entry for those wishing to employ pre-annotation on similar tasks.

The second row in Table 3.1a shows a similar positive trend for the combination of pre-annotation and correction propagation, but with weaker significance. It is unclear whether this trend is explained entirely by the presence of pre-annotation, or whether correction propagation is playing a role in helping or hurting accuracy. The third row of Table 3.1a shows mixed signs with no statistical significance, preventing us from drawing any strong conclusions about the effect of correction propagation above and beyond that of pre-annotation.

The first row in Table 3.1b shows the difference in the mean time required to label words with and without pre-annotations. Pre-annotations generated by models of quality 80% or better decrease average word annotation time by around 10-20 seconds, and that decrease is usually greater than can be explained by the variance in the data, although this trend is still noisy in our current data. Since most words take between 10 and 70 seconds to annotate (see Figure 3.1b), 10-20 seconds is an appreciable improvement. Pending additional evidence to strengthen the outcome, it is reasonably clear that moderately good pre-annotation reduce the time required for annotation.

One natural way to attempt to anticipate the effect of additional data is to group data points from similar annotation conditions. In Table 3.2 we do this and test our null hypotheses again. It is worth noting that the results in Table 3.2 are less applicable to most real world annotation situations than Table 3.1, since they involve comparing the times and accuracies of words annotated with no pre-annotations (the none case) with the times and accuracies of words annotated with a mixture of two different models. However, since the models being mixed are those of similar quality, these results should give us an idea of what our data will look like if present trends continue.

The trends that we noted in Table 3.1 are slightly clearer in Table 3.2: both pre-annotation and the combination of pre-annotation and correction propagation reduce annotation time and increase annotation accuracy using low-to-medium quality pre-annotation models. Again, the individual contribution of correction propagation is unclear, although there is some indication in the third row of Table 3.2b that it may negatively impact annotation speed. It seems safe to say that whether it hurts or helps, the effects of correction propagation on annotator speed and accuracy are dwarfed by the effects of pre-annotation.

Because machine learners improve as additional annotations become available, annotators in large projects will often have access to high quality machine assistance, making the effects of high quality assistance of particular interest. Accordingly we asked each participant in the study to

(a) Change in Mean Word Accuracy

Control	**Test**	**Pre-annotation model quality (versus gold standard)**						
Condition	**Condition**	25 & 36	36 & 47	47 & 58	58 & 68	68 & 79	79 & 90	90 & 100
“none”	PL	+1.5	+1.4	**+4.1**	<u>**+5.1**</u>	<u>**+4.7**</u>	<u>**+5.2**</u>	<u>**+6.8**</u>
“none”	PL+CP	+3.0	+2.3	+1.9	+2.9	**+4.2**	<u>**+5.1**</u>	<u>**+5.4**</u>
PL	PL+CP	+1.4	+0.9	-2.3	-2.3	-0.5	-0.2	-1.5

(b) Change in Mean Word Time (sec)

Control	**Test**	**Pre-annotation model quality (versus gold standard)**						
Condition	**Condition**	25 & 36	36 & 47	47 & 58	58 & 68	68 & 79	79 & 90	90 & 100
“none”	PL	-1.8	+0.8	+13.6	+4.9	**-12.9**	**-14.9**	<u>**-19.2**</u>
“none”	PL+CP	-1.1	+4.1	-2.5	-9.6	-7.9	-3.4	-2.9
PL	PL+CP	+0.7	+3.3	-16.1	-14.5	+5.0	+11.6	**+16.3**

Table 3.2: Identical to Table 3.1 after grouping observations more coarsely in order to account for current data scarcity.

annotate two additional randomly selected sentences using what we anticipated would be the most effective experimental condition: 90+C. This yielded an additional 122 word level data points. Adding this new data to Table 3.1 left the mean accuracy difference between none and 90+C unchanged, but changed the mean time difference from -2.9 to -19.0 seconds, and that difference was highly statistically significant. It is likely that additional data would similarly strengthen our other results.

3.5 Conclusions and Future Work

We have presented a systematic evaluation of pre-annotation and correction propagation together in a controlled user study, providing a detailed data point for those wishing to apply these techniques to similar domains. Preliminary analysis indicates that for our experimental setup, even low quality pre-annotations are effective in increasing average annotator accuracy (i.e. agreement with a gold standard) by 5-7%. Our results also indicate that pre-annotations of moderate quality reduce average annotation time by 10-20 seconds per word. Correction propagation’s contribution to annotator speed and accuracy is unclear.

This preliminary analysis will inform continuing work on the creation of the Syriac Electronic Corpus, described in Section 3.1. As a part of this, we plan to conduct additional analyses of the study's timing data to identify ways of improving the efficiency of user interactions in our GUI. Additionally, we plan to use the timing data collected during the course of the study to model the cost of Syriac morphological annotation so that cost-conscious active learning may be used to reduce the cost of learning high quality pre-annotation models [22]. Although active learning shows theoretical promise, there is still a large need for evidence that it can reduce cost in a practical setting.

Chapter 4

Improving Annotation Efficiency in Under-Resourced Languages using Bayesian Data Analysis

Author List

Paul Felt, Eric Ringger, Kevin Seppi, Kristian Heal, Robbie Haertel, Deryle Lonsdale

Submission Venue

This paper has been submitted to the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP). It has been de-anonymized for inclusion in this thesis.

MS Thesis Context

This paper fits into the larger thesis by continuing the analysis of the timing data gathered in Chapter 3, gaining more clarity about the effects of correction propagation, and distilling some additional insights from the data. This paper provides guidance to practitioners wishing to improve the efficiency of their annotation process by gaining insights from limited amounts of user interaction data. In this way it contributes to the larger thesis's goal of improving evaluation methods for machine assistance techniques.

Abstract

Manual annotation of large textual corpora can be cost-prohibitive, especially for rare and under-resourced languages. It is therefore critical to make the annotation process as efficient as possible.

User interaction data can shed light on inefficiencies in the annotation process; however, user interaction data can be limited and costly to obtain in under-resourced domains. We advocate a Bayesian approach to analyzing user interaction data, since it is amenable to multiple rounds of data collection, and lends itself to jointly inferring many parameters of interest. We validate this position by conducting a Bayesian analysis of data from a previously reported user study in which participants annotated Syriac text with morphological analyses. The Bayesian analysis improves on a previous, simpler analysis by identifying two important inefficiencies in the annotation process.

4.1 Introduction

Manual annotation of large textual corpora can be cost-prohibitive, especially for rare and under-resourced languages. Developers of annotated corpora often attempt to improve annotation efficiency by developing intuitive annotation interfaces or by offering annotators some form of automatic assistance. Each decision or variable affecting the annotator work environment can potentially increase or decrease annotator speed and/or accuracy. Understanding these effects is prerequisite to improving annotation efficiency and managing the cost of corpus creation. Although promising decisions may be identified by intuition or using simulations, the best source of information about these effects is actual user interaction data: a record of annotator actions and the time it took to complete each action. Thus, the problem of improving annotation efficiency is highly dependant on effective user interaction data analysis.

Traditionally, developers of annotated corpora have used simple comparisons of mean time and accuracy to evaluate important variables affecting the annotation process. That is, they have monitored the performance of annotators with and without the presence of some variable and afterwards compared the two groups' average performance. These analyses are appealing since they are so simple and intuitive. However, there are some important problems with this approach. For one thing, data must be collected such that confounding variables are distributed evenly between the cases being compared. This requirement limits the number of questions that may be answered using a single data sample. Additionally, if mean comparisons are being backed by statistical hypothesis tests, ways of legitimately gathering and analyzing the data are further limited (see Section 4.4). In under-resourced domains where user interaction data is difficult to come by, these considerations severely limit the number of questions that can be answered by comparing means.

Bayesian data analysis overcomes many of the shortcomings associated with simple mean comparison analyes, and thus is well suited to solving the larger problem of using modest amounts of user interaction data to generate insights that lead to improving annotation efficiency. We support this claim using a concrete example. We have previously reported on a user study in which we gathered user interaction data, and then used mean value comparisons and hypothesis testing to answer the question of when two particular annotation assistance techniques, pre-annotation and correction propagation, improve the speed and/or accuracy of annotators engaged in Syriac morphological analysis [16]. However, our results with respect to correction propagation were inconclusive, and our ability to answer additional questions using the same methodology was limited (see Section 4.4). A Bayesian approach allows us to incorporate additional data, jointly estimate the effects of many variables, and thus explicitly account for confounding effects. Our new results shed light on the helpfulness of correction propagation, and help to identify two important inefficiencies in our annotation process.

Because of space considerations, we analyze variable effects only with respect to annotator time, deferring a similar analysis with respect to annotator accuracy to future work. To the best of our knowledge, Bayesian data analysis has not previously been applied to the problem of improving annotation efficiency, possibly for lack of a relevant detailed example. For this reason, this paper attempts to be transparent in all the details of a Bayesian data analysis, explaining details that statistical analyses would typically only refer to in passing. We encourage readers to bear in mind that this detailed analysis serves the larger purpose of facilitating the efforts of future corpora builders to improve their annotation efficiency by conducting similar analyses of their own user interaction data.

Section 4.2 describes pre-annotation, correction propagation, and previous work analyzing their effects. Section 4.3 describes the project motivating this work and the conditions under which the data were gathered. Section 4.4 analyzes the data using traditional mean comparisons and hypothesis testing. Section 4.5 analyzes the data using a Bayesian approach. Section 4.6 discusses our conclusions and outlines future work.

4.2 Related Work

Mean value comparisons have been used too extensively in analyzing user interaction data to make a complete listing feasible. Instead, we describe experiments that have used mean value comparisons to analyze two variables that are particularly important in our data: pre-annotation and correction propagation.

Pre-annotation involves asking human annotators to correct sentences that have already been automatically pre-annotated using a learned model. Pre-annotation has the potential to reduce annotation time if correcting automatically proposed annotations is easier than annotating from scratch. Previous work has used mean value comparisons to show that pre-annotation increases the accuracy and speed of annotators engaged in English Part-of-speech tagging [29], Chinese treebank annotation [9], parsing [1, 5], and named entity recognition [18].

Correction propagation is a way of potentially improving pre-annotation quality by automatically revising downstream pre-annotations as a human annotator makes corrections to upstream pre-annotations [27]. Correction propagation has been shown to reduce the mean number of user interactions required to complete an information extraction annotation task in a hypothetical graphical user interface. This analysis was carried out in simulation rather than using actual user interaction data.

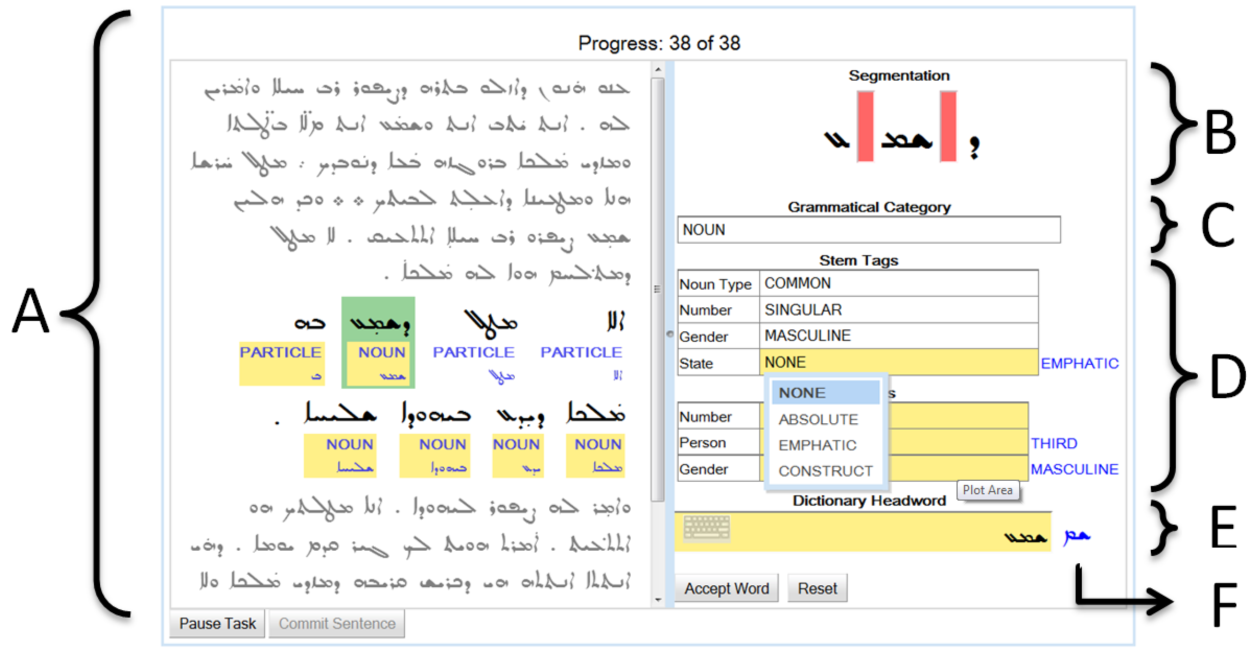
All previous work of which we are aware analyzes pre-annotation and correction propagation only at state-of-the-art qualities. This is insufficient for under-resourced domains in which high quality predictive models are not available, since it is conceivable that poor predictions could be distracting and actually reduce annotation speed.

4.3 The Data

4.3.1 Syriac User Study

In Felt et al. [16], we describe a controlled, web-mediated user study used to collect data about annotators engaged in Syriac morphological analysis. Only details necessary to understanding the current analysis are repeated here.

In the user study, participants received training, completed four practice sentences, then were monitored as they annotated 30 sentences. For each sentence they worked on, annotators were randomly assigned a different experimental condition (see Section 4.3.3). Nine annotators



Grammatical Category: NOUN

Stem Tags

Noun Type	COMMON	
Number	SINGULAR	
Gender	MASCULINE	
State	NONE	EMPHATIC

NONE / ABSOLUTE / EMPHATIC / CONSTRUCT

Number		
Person		THIRD
Gender		MASCULINE

Figure 4.1: The GUI used for Syriac morphological analysis. The sentence being annotated and its context is shown in A. In B a prefix and suffix are identified. A grammatical category is selected in C, and then additional tags relevant to the grammatical category are selected in D. In E the word is linked to a dictionary form. Correction propagation hyperlinks appear to the right of their corresponding fields (F).

completed the study, each annotating 30 sentences, or 152 words, resulting in a total of 1,368 word-level data points. The 30 user study sentences were selected from The Acts of Judas Thomas, an apocryphal text that is similar to the New Testament [48].

Annotators worked in a web application implemented inside of CCASH,[1] which automatically collects user interaction data. The graphical annotation interface is shown in Figure 4.1. Words that are pre-annotated appear to annotators with the fields already filled out. When correction propagation is active, updated model guesses appear as hyperlinks to the right of their corresponding field (Figure 4.1F). Updated model guesses that apply to other words in the sentence replace previous pre-annotation values without requiring any user interaction.

CCASH reported the amount of time in fractional seconds that each word was in focus. We use these times to represent the amount of time required to annotate each word. This time is not a

[1]http://ccash.sourceforge.net

perfect stand-in for the time an annotator spent actually working on each word. An annotator may spend time considering a word that is not actually selected. Also, the first word of each sentence will naturally tend to be selected longer than other words in the sentence as an annotator orients herself by reading the sentence and context. However, these times should be an acceptable approximation of the true time spent annotating each word. Analyses are conducted on a word level rather than a sentence level so that annotation times may be easily compared without having to choose sentences of uniform length for the study.

4.3.2 Pre-annotation Models

The model used to generate pre-annotations for this study was Syromorph, a probabilistic data-driven Syriac morphological analyzer developed by McClanahan et al. [30]. Syromorph models were trained on random subsets of the morphologically annotated Syriac Peshitta New Testament compiled by Kiraz [25], and evaluated against a gold standard annotation set constructed from the 30 user study sentences. The gold standard was constructed by two expert Syriac linguists who independently labeled each sentence, then discussed and resolved all disagreements in their labelings. All accuracies mentioned in this paper are calculated against this gold standard.

4.3.3 Experimental Conditions

Pre-annotations were supplied to annotators at the following accuracy levels: $\{$*none*, 25%, 35%, 45%, 55%, 65%, 75%, 90%, 100%$\}$. In the *none* case, no pre-annotations were given. In the 100% case, gold standard annotations were used. In all intermediate cases, Syromorph models trained to the target accuracy provided pre-annotations. Note that correction propagation requires a supporting pre-annotation model; consequently it cannot be applied to the *none* or 100% cases. In all, this design yields $|\{none, 100\}| + |\{25, 36, 47, 58, 68, 79, 90\} \times \{+CP, -CP\}|$ or 16 parameter combinations to test. We refer to each combination as an *experimental condition*. Annotators were assigned to experimental conditions in such a way that they encountered each experimental condition in random order and in approximately equal proportions. We also ensured that each

	25	36	47	58	68	79	90	100%
-CP	5.4	-9.9	11.1	15.4	-7.1	**-20.0**	-10.4	**-27.6**
+CP	-7.0	5.0	3.1	-8.0	-11.2	-3.9	-2.9	NA

Table 4.1: Simple analysis. This table's cells contain the difference between the mean time required to annotate words under a given experimental condition and the mean time required to annotate words without any assistance (the *none* condition). Statistical significance at or below the 0.05 level according to a double-sided *t*-test is indicated by bolding.

sentence was annotated under each experimental condition approximately the same number of times so that the data does not unduly favor the idiosyncrasies of one sentence over those of another. This design caused our 1,368 data points to be divided roughly evenly among the 16 experimental conditions, giving roughly 85 data points per condition.

4.4 Simple Analysis

Because we went to considerable lengths while gathering data to ensure that confounding effects were distributed evenly across experimental conditions, we were previously able to estimate the effect of each experimental condition by simply comparing the mean annotation times of words collected under each experimental condition with the mean value of the data points collected under the *none* condition [16]. For convenience to the reader, this analysis is reproduced in Table 4.1. Each of the 15 comparisons is tested for statistical significance at the 0.05 level using a standard double-sided Student's *t*-test. Table 4.1 indicates that when pre-annotations are about 80% accurate or better, they significantly increase annotation speed. It is unclear whether using correction propagation in addition to pre-annotation is helpful or harmful.

Now we would like to extend this analysis to include some additional data described in Section 4.5.1. However, the hypothesis testing framework does not allow us to easily incorporate data that was not part of the original experiment design. To see this, recall that p-values depend on the likelihood function of the data assuming the null hypothesis is true. This likelihood function, in turn, depends on the experimental design under which the data was collected [4].

We would also like to estimate the effects of other variables on annotation time, including who did each annotation, whether they clicked on a correction propagation hyperlink, the word's grammatical category, and so on. These factors are interesting both in their own right and also as potential sources of confusion in the data whenever they are not sufficiently averaged over. For example, because only 9 annotators completed the study, effects associated with annotator identity such as average annotator speed and internet latency are distributed somewhat unevenly in the experimental condition data. However, if we group the data by some arbitrary attribute and compare means, it is very likely that our results will be biased by confounding effects, since no attempt was made while gathering the data to ensure that confounding effects were evenly distributed across any attribute other than experimental condition.

4.5 Bayesian Analysis

A Bayesian approach can help us overcome many of the problems described in the previous section. Bayesian methodologies allow us to incorporate additional data into our analysis without worrying about whether it was obtained all at once or sequentially [4]. Also, using standard Bayesian machinery, we can propose a model of our data and jointly infer distributions over numerous variables of interest, explicitly accounting for potentially confounding effects.

Conducting a parametric Bayesian Analysis involves constructing a parametric model of the likelihood of the observed data, setting a prior over each unobserved parameter in that model, and then using standard Bayesian machinery (e.g. Gibb's sampling) to infer posterior distributions over each parameter, given the data [20]. We proceed by describing our data in Section 4.5.1, defining a likelihood distribution in Section 4.5.2, setting priors over each parameter in our model in Section 4.5.3, sampling from the posterior in Section 4.5.4, and finally analyzing our posteriors in Section 4.5.5.

4.5.1 Data

We use the data described in Section 4.3 as well as some data gathered after the main study. Each of the 9 participants additionally annotated two randomly selected sentences, using 90% accurate pre-annotations and correction propagation, for a total of 122 extra words. Finally, one participant annotated an extra 10 sentences under randomly assigned experimental conditions for a total of 47 extra words. We now have a total of 1,537 data points. Each data point consists of the time taken to annotate each word along with additional information such as who did the annotation, what degree of assistance she received, and so forth.

4.5.2 Likelihood

We begin by proposing a probability distribution over our data $\boldsymbol{y}$. Because we are dealing with time, it is not unreasonable to imagine that we start with some average work cost κ and then add or subtract from that cost based on who is annotating, under what experimental condition they are annotating, what kind of word is being annotated, etc. If we finish by adding symmetric noise then this may be modeled as a normal distribution. Note that a normal likelihood is not a perfect choice, since it allocates some probability for words with negative times, but it is convenient and seems to work well in practice. We accordingly model the density of a single observation y_{hatbro} as a normally distributed sum of additive effects, each with a subscript corresponding to the h,a,t,b,r, or o subscripts on y. We model time effects:

- θ_h: Of the identity of the annotator. There are nine $\boldsymbol{\theta}$ variables, corresponding to each annotator in the study.
- α_a: Of the experimental condition under which the current word's sentence was annotated. There are 16 $\boldsymbol{\alpha}$ variables.
- τ_t: Of the grammatical category of the word (noun, verb, etc). There are 6 $\boldsymbol{\tau}$ variables corresponding to the six grammatical categories used in the data.

- β_b: Of the position of the word in the sentence. We combine all word positions after 3, so there are only four $\boldsymbol{\beta}$ variables.
- ρ_r: Of correction propagation hyperlinks being shown or not. There are 2 $\boldsymbol{\rho}$ variables.
- ω_o Of correction propagation hyperlinks being clicked or not. There are 2 $\boldsymbol{\omega}$ variables.
- κ A time offset common to all words.
- σ^2 A time variance common to all words.

The density of a single observation is then

$$\begin{aligned}&y_{hatbro}|\theta_h, \alpha_a, \tau_t, \beta_b, \rho_r, \omega_o, \kappa, \sigma^2\\ \sim &N(\theta_h + \alpha_a + \tau_t + \beta_b + \rho_r + \omega_o + \kappa, \sigma^2)\end{aligned}$$

The likelihood of our data set is obtained by taking the product of the density of each observation.

$$\begin{aligned}&L(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{\beta}, \boldsymbol{\rho}, \boldsymbol{\omega}, \kappa, \sigma^2)\\ =&\prod_{\mathbf{y}} p(y_{hatbro}|\theta_h, \alpha_a, \tau_t, \beta_b, \rho_r, \omega_o, \kappa, \sigma^2)\\ =&(2\pi\sigma^2)^{-\frac{N}{2}} e^{-\frac{1}{2\sigma^2}\sum_{\mathbf{y}}(y_{hatbro}-\theta_h-\alpha_a-\tau_t-\beta_b-\rho_r-\omega_o-\kappa)^2}\end{aligned}$$

4.5.3 Priors

We determined our priors as follows. An expert annotating data outside of the user study took around 90 seconds per word, and his times varied by as much as a minute. We use this to inform our priors over κ and σ^2. $\kappa \sim N(90, 50/3)$, allowing our offset to vary as much as 50 seconds on either side. Because σ^2 cannot be negative, we model it using a Gamma parameterized by shape and scale. If σ^2 were equal to 2500, that would allow annotation times to vary by about $3 * \sqrt{2500} = 75$

seconds on either side. We therefore let $\sigma^2 \sim Gamma(50, 50)$ which has a mean value of 2500 with a fair amount of spread, accurately reflecting our uncertainty about this quantity.

We model α_a, θ_h, β_b, τ_r, ρ_r, and ω_o as normally distributed contributions centered around 0, so as to have no effect by default. We would like to see how the data shapes their posteriors, so we set their priors to be relatively uninformative normal distributions $N(0, \frac{40}{3})$, allowing the effect to have mass between -40 and +40. To be clear, negative effect values mean that an effect reduces the total annotation time, and positive effect values mean that an effect increases the total annotation time.

4.5.4 Sampling

Gibbs sampling may be used to obtain samples from the joint posterior provided we can sample from the complete conditional distribution of each variable [20]. We derive the complete conditional distributions for each parameter by writing out the posterior and then keeping all terms pertaining to the parameter in question (the complete derivation may be found in Chapter 9).

Let $g(\mathbf{y}) = ln(L(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{\beta}, \boldsymbol{\rho}, \boldsymbol{\omega}, \kappa, \sigma^2))$. Also let c represent the constant that ensures a proper distribution. Finally, let $\boldsymbol{y}_a$ indicate the set of data that were annotated under condition a, $\boldsymbol{y}_h$ be the set of data that were annotated by annotator h, and so on. Then the logarithm of the complete conditionals are as follows:

$$[\theta_h] = -\frac{\theta_h^2}{2(\frac{40}{3})^2} + g(\boldsymbol{y}_h) + c$$

$$[\alpha_a] = -\frac{\alpha_a^2}{2(\frac{40}{3})^2} + g(\boldsymbol{y}_a) + c$$

$$[\tau_t] = -\frac{\tau_t^2}{2(\frac{40}{3})^2} + g(\boldsymbol{y}_t) + c$$

$$[\beta_b] = -\frac{\beta_b^2}{2(\frac{40}{3})^2} + g(\boldsymbol{y}_b) + c$$

$$[\rho_r] = -\frac{\rho_r^2}{2(\frac{40}{3})^2} + g(\boldsymbol{y}_r) + c$$

$$[\omega_o] = -\frac{\omega_o^2}{2(\frac{40}{3})^2} + g(\boldsymbol{y}_o) + c$$

$$[\kappa] = -\frac{(\kappa - 90)^2}{2(\frac{50}{3})^2} + g(\boldsymbol{y}) + c$$

$$\left[\sigma^2\right] = (49 - \frac{N}{2})ln(\sigma^2) - \frac{\sigma^2}{50} + g(\boldsymbol{y}) + c$$

Although we cannot reduce these conditionals to a convenient closed form that we know how to sample from, we can use use Metropolis sampling to obtain samples from each conditional inside the Gibbs sampling loop [20]. Sampling begins with a “burn-in” period during which samples are discarded as the sampling chain moves away from arbitrary starting values and settles into higher probability regions of the probability surface. After burning in for about 10,000 rounds, we obtain 500,000 samples and then thin by a factor of 50, leaving us with 10,000 samples from our joint posterior.

Trace plots charting the sampled values of each parameter show good coverage of the variables, providing some evidence that our sampling chain has converged. We also test our chain for convergence using the Raftery-Lewis diagnostic, which measures the movement of various parameter percentile values across the chain, and also the Geweke diagnostic, which measures the movement of parameter means across the chain [11]. The chain’s Raftery-Lewis IRL measures are all below the rule-of-thumb of 5, and all but two of the Geweke z-scores have an an absolute value of less than the rule-of-thumb of 1.96, giving additional evidence that our chain has successfully converged.

	25	36	47	58	68	79	90	100%
-CP	5.6	-1.9	10.8	7.9	-3.2	**-14.7**	-7.1	**-20.0**
+CP	-13.8	-3.7	-6.3	-13.0	-12.3	-10.8	**-20.2**	NA

Table 4.2: The posterior distribution over $\boldsymbol{\alpha}$ discovered during the Bayesian analysis corroborates the results of the simple analysis in Table 4.1. This table's cells contain the mean difference between each parameter α_k (the seconds-per-word time contribution of annotation condition k) and α_0 (the *none* condition). Conditions whose values are greater or less than α_0 with greater than 0.95 probability are bolded.

4.5.5 Posteriors and Discussion

We now have 10,000 samples from the joint posterior distribution, where each sample specifies a value for each parameter in our model. By inspecting the values corresponding to parameters of interest and ignoring the rest, we obtain marginal posterior probability distributions over particular parameters. This allows us to answer questions related to how each factor we have modeled affects annotation speed. This section proceeds by first revisiting the question we addressed in Section 4.4, and afterwards examining the effects of additional parameters.

We can answer the question of how pre-annotation and correction propagation affect annotation speed by inspecting the values of $\boldsymbol{\alpha}$ in our joint posterior samples. Table 4.2 shows the average difference between the values of each experimental condition parameter α_a and the *none* condition parameter α_0. By comparing Table 4.2 with Table 4.1, we can get a feel for how controlling for confounding effects affects our conclusions. The effects of pre-annotation are nearly the same as before, giving us increased confidence in our conclusion that pre-annotations increase annotation speed when they are at least 80% accurate. Correction propagation, however, looks far more promising than it did before. Our explanation for these differences is that the Bayesian analysis separately accounts for three effects related to our implementation of correction propagation: the $\boldsymbol{\omega}$ variables account for the effect of hyperlinks appearing, the $\boldsymbol{\rho}$ variables account for the effect of hyperlinks being clicked, and the $\boldsymbol{\alpha}$ variables for which correction propagation is active account for the effect of corrections being made (without hyperlinks) to words not currently in focus.

Since the results in Table 4.1 and Table 4.2 are very similar, one might be inclined to prefer hypothesis testing to a Bayesian analysis because it appears to involve less work. However, recall that we are only able to do the simple analysis in Section 4.4 because considerable effort was put into gathering data. Language resource developers in under-resourced domains will seldom have the luxury of collecting redundant user interaction data in a highly controlled setting. A Bayesian analysis can be appropriately applied to user interaction data collected in the wild as a natural by-product of a project's progression, and used iteratively to track trends over time. Priors may even be updated in light of previous analyses as long as the data used in those analyses are not also re-used.

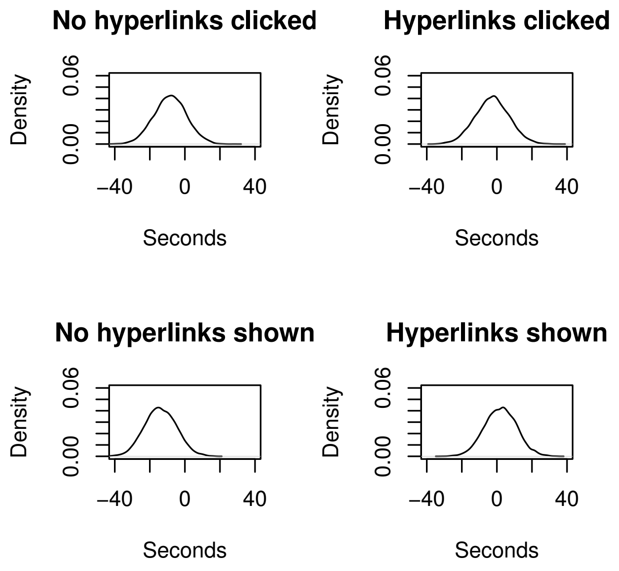


Figure 4.2: The marginal posteriors of each ρ_r (top row) and ω_o (bottom row). These values indicate the time contribution that having correction propagation hyperlinks be shown or clicked on lends to a word's overall cost. Negative values indicate that they decrease the base cost. Positive values indicate that they increase the base cost.

The marginal posteriors of $\boldsymbol{\rho}$ and $\boldsymbol{\omega}$ in Figure 4.2 indicate that when correction propagation hyperlinks are shown, words tend to take more time to annotate than when hyperlinks are not shown, and similarly when hyperlinks are clicked, words take more time to annotate than when no hyperlinks are clicked. Although these differences are not significant using 95% credible intervals, they provide evidence that using the mechanism of hyperlinks to implement correction propagation incurs a measurable cost. One possible explanation for this is that the appearance of hyperlinks is a distraction, causing annotators to pause while they assimilate new visual information. Given the modest potential gains afforded by correction propagation (see Table 4.2), it appears clear in retrospect that using hyperlinks to signal correction propagation updates for the word in focus was an inappropriately expensive visual mechanism.

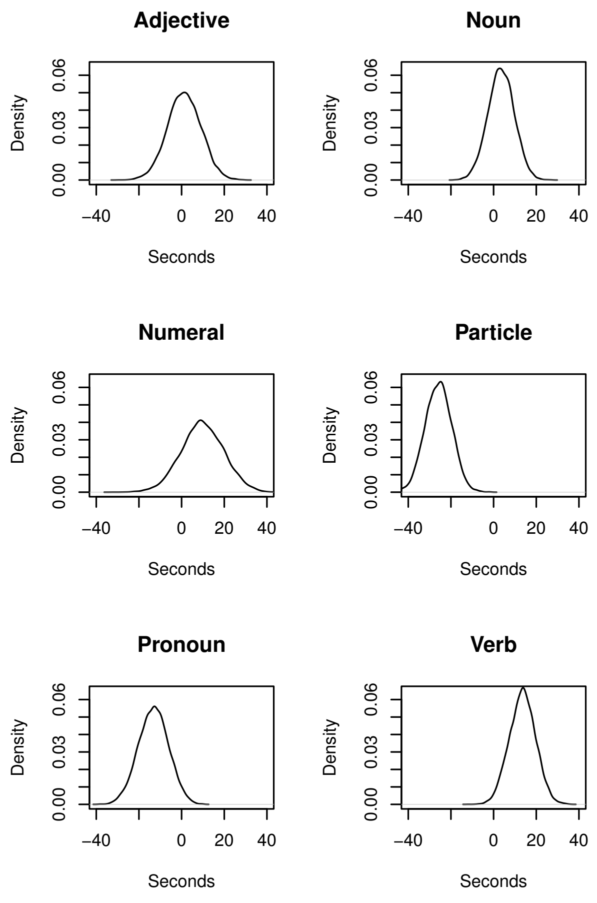


Figure 4.3: The marginal posteriors of each τ_t. These values indicate the time contribution that belonging to a certain grammatical category lends to a word's overall cost. Negative values indicate that they decrease the base cost. Positive values indicate that they increase the base cost.

The marginal posteriors of the $\boldsymbol{\tau}$ parameters in Figure 4.3 estimate the time contributions of different grammatical categories of words. Most of these values are unsurprising. The fact that particles tend to be inexpensive and verbs tend to be costly is what we would expect. However, the fact that numerals are tending to be costly is surprising. Collaborating domain experts tell us that labeling numerals ought to be relatively straight-forward, so this data indicates that there is some ambiguity in the labeling documentation or interface that we can address to improve the efficiency of our annotators. Using this insight, we were able to filter the free-form participant feedback and prioritize participant suggestions for improving numeral training.

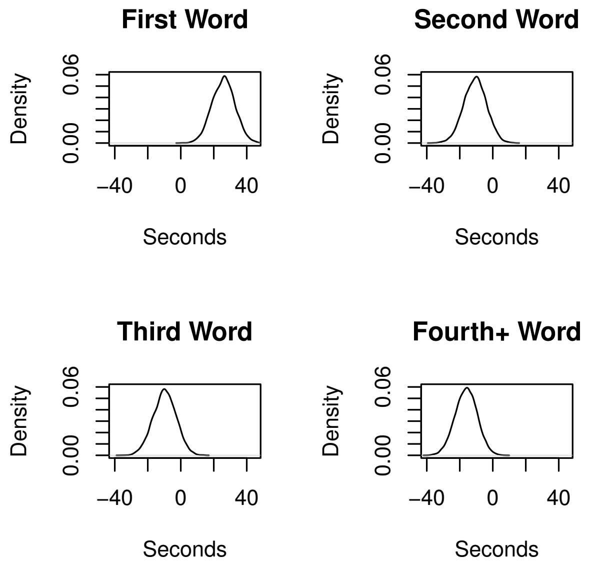


Figure 4.4: The marginal posteriors of each β_b. These values indicate the time contribution that being in a particular position in the sentence lends to a word's overall cost. Negative values indicate that they decrease the base cost. Positive values indicate that they increase the base cost.

The $\boldsymbol{\beta}$ parameters in Figure 4.4 show that the first word of each sentence incurs a significant time penalty as annotators orient themselves within the sentence. This outcome was anticipated when we designed our model, and should help account for some of the variance in the data. However, it was unclear before the analysis whether to expect that the second and third words would also participate in this "learning curve" effect. It appears that after the first word in the sentence, all other word positions incur approximately the same cost.

One additional benefit of having a Bayesian model of the data is that we can use it to generate samples from the posterior predictive distribution: the distribution over values of future data given our priors, model, and the data we have observed so far. Sampling from the posterior predictive allows us to use our model to predict what future data will look like. This is done by using the parameter values we have already sampled from the model's joint posterior distribution to draw data observations from our likelihood function. Additionally, by selecting posterior predictive samples in which certain conditions hold, we can draw a distribution over predicted future observations given those conditions.

The posterior predictive distribution can be useful in estimating future annotator performance, and could also have applications in the realm of cost-consious active learning, in which one wishes to choose additional data to annotate based on their expected value as well as their expected cost [22]. For example, by selecting the posterior predictive data in which annotator #2 (a particularly fast annotator) is annotating verbs, we estimate that annotator #2 will take on average 26 seconds to annotate each verb. On the other hand, we estimate that annotator #9 (a slower annotator) will take on average 45 seconds to annotate verbs.

4.6 Conclusions and Future Work

We have used a Bayesian analysis of limited user interaction data to strengthen our previous conclusion that pre-annotations that are at least approximately 80% accurate increase the speed of Syriac morphological analysis. The analysis has also provided evidence that the hyperlink mechanism we used to indicate correction propagations for the in-focus word is inappropriately expensive, and that correction propagation for words other than the word currently in focus is much more promising than we previously thought. In addition, the Bayesian analysis shows that there is an inordinate amount of inefficiency associated with the "Numeric" grammatical category. In retrospect, this seems likely to be due to insufficient training material.

We have demonstrated conducting a Bayesian data analysis to jointly estimate various parameters of interest from limited user interaction data. Our work provides a detailed methodological data point for others wishing to analyze user interaction data in under-resourced language domains. More broadly, by demonstrating an effective approach to analyzing sparse user interaction data, we have given developers of annotated corpora in under-resourced domains tools that they need in order to improve the efficiency of their own annotation processes.

In the future we plan to perform a similar analysis on annotator accuracy. Our results will be used to improve the efficiency of annotators engaged in creating the Syriac Electronic Corpus. We plan to use a similar methodologies to evaluate the performance of other machine assistance techniques in speeding up Syriac morphological analysis, including cost conscious active learning [22].

Chapter 5

Conclusions

This thesis makes a contribution to the field of annotated corpus development by providing tools and demonstrating a methodology for empirically evaluating machine-assisted annotation techniques. This thesis additionally contributes to the field of Syriac studies by empirically quantifying the effectiveness of pre-annotation and correction propagation in improving the speed and accuracy of annotators engaged in Syriac morphological analysis.

We designed and built CCASH, an online linguistic annotation system well-suited to running instrumented user studies involving machine assistance, and made it available to the community under an open-source license. We employed CCASH to gather user interaction data from annotators engaged in Syriac morphological analysis with and without two forms of machine assistance: pre-annotation and correction propagation.

We conducted a traditional analysis of the data by comparing the mean times and accuracies of annotators working with various levels of machine assistance. This simple analysis showed that pre-annotations that are at least 60% accurate increase annotator accuracy by 5-7%, and also that pre-annotations that are at least 80% accurate increase annotator speed by 10-20 seconds per word. The role of correction propagation using this simple analysis was unclear. We additionally conducted a Bayesian analysis of annotator time, modeling the effect of pre-annotation and correction propagation as well as a number of potentially confounding variables. The Bayesian analysis strengthened the results of our simple analysis, and also shed light on the effect that correction propagation has on annotator speed. Specifically, in the Bayesian analysis correction propagation appears to increase annotator speed slightly. However, the mechanism of showing correction propagation updates as hyperlinks for the word currently in focus incurs a large enough time penalty to potentially counteract any good that correction propagation might be doing elsewhere in the sentence. This insight will allow us to improve the performance of correction propagation going forward. We also found that words in the "Numerical" grammatical category incur far more cost than they should. This is an insight we can use to improve annotation training.

The knowledge gained from these analyses will allow us to reduce the annotation cost required to create the Syriac Electronic Corpus. It may also help inform the decisions of corpora developers working on similar tasks, particularly in low resource domains where machine assistance quality is limited. More broadly, the tools and methodology demonstrated in this thesis may be used as resources by those interested in answering similar questions in different annotation domains and for different modes of machine assistance.

In the future we plan to evaluate the effectiveness of additional machine-assisted annotation techniques, including various kinds of active learning [41]. Active learning involves selecting instances for human annotation that are likely to be most informative in training an automatic annotator. Because a high quality automatic annotator can be used to reduce annotation cost (e.g. through pre-annotations), active learning has the potential to reduce the cost necessary to label a corpus from scratch. However, active learning is not widely used in practice, partly because it has not been shown empirically to reduce cost in enough practical settings [44].

Chapter 6

Appendix A: CCASH Documentation

CCASH's documentation is subject to change based on user needs and feedback. The most up-to-date version of this documentation may be found at `http://facwiki.cs.byu.edu/nlp/index.php/CCASH`.

6.1 Introduction

6.1.1 What is CCASH?

CCASH (Cost-Conscious Annotation Supervised by Humans) is a web-based annotation framework. It is designed to be an environment for evaluating state-of-the-art and experimental techniques for efficient annotation and also for applying those techniques to real world annotation projects. While designing CCASH we had our eye particularly on Active Learning; however other techniques such as feature labeling and incorporating rich prior knowledge could also be incorporated into CCASH without too much trouble.

6.1.2 How does it work?

CCASH coordinates the activities of two components:

- **Annotation Tasks** are graphical user interface that run in your browser and allow you to annotate instances, or correct automatic annotations. An annotation task's job is to display a particular kind of instance and solicit a particular kind of annotation. CCASH tasks are implemented with the Google Web Toolkit, allowing you to write code in Java assisted by GWT's WYSIWYG editors.

- **Annotation Managers** run as xmlrpc services on the network. As such they may be written in any language with an xmlrpc implementation (that is to say, almost anything). Annotation managers are in charge of two important tasks:

    1. Provide annotators with an optionally pre-annotated instances
    2. Record annotations

In a typical annotation scenario, CCASH would query an annotation manager for a pre-annotated instance, then present that instance to a human annotator via a compatible GUI task. After the annotator finished, the completed annotation would be sent back to the annotation manager to be preserved.

6.2 Getting Started

Eclipse

CCASH is an eclipse project, so you will want to get a current copy of the eclipse IDE for Java EE developers (http://www.eclipse.org/downloads). You'll additionally need to install the following eclipse plugins:

- The Google Plugin for Eclipse (http://code.google.com/eclipse), along with the Google Web Toolkit SDK.
- Subversive(http://www.eclipse.org/subversive) or Subclipse(http://subclipse.tigris.org) in order to use Subversion in Eclipse.

Postgres

CCASH manages its data in a relational database. We chose postgres as the default implementation because of its permissive licensing and sub-second timing values. You will need to install the postgres server on your system. Mysql also works if you prefer.

Configure

After installing postgres, you must configure postgres to accept connections from your CCASH install. Do this by editing the pg_hba.conf file and changing the line that reads

```
host    all         all         127.0.0.1/32         ident
```

```
host    all         all         127.0.0.1/32         trust
```

This tells postgres to trust all connections from the localhost. This is fine for development. (In the future when you deploy Ccash, you will probably want to increase security by changing the word "trust" to "md5" which will require you to create a postgres account and password for CCASH. The username and password can be whatever you want as long as you change the corresponding data inside of the file Ccash/src/META-INF/persistence.xml).

Create a database

Create a postgres database for CCASH by running the following command:

```
createdb -U postgres ccash
```

Create a database user

Create a postgres database for CCASH by running the following command:

```
createuser -U postgres ccash
```

Get CCASH

For a copy of CCASH licensed under the AGPL, use Subclipse to check out the code at https://ccash.svn.sourceforg
If you are interested in CCASH under a different license, please contact us directly.

Run CCASH

To run CCASH, right-click on the eclipse CCASH project, click “Run As,” and select “Web Application.” After a minute a “Development Mode” tab will open in Eclipse and display a url. Copy this url into a browser, and you will see the CCASH login screen. Login with username “admin” and password “passwd99”. You can change this password after logging in by clicking the “Admin” menu item, and selecting “Annotators”.

“Hello, World” annotation task

Start annotating by following the tutorial at `https://facwiki.cs.byu.edu/nlp/index.php/Doing_Simple_Sentiment_Classification`.

6.3 Custom tasks

CCASH is an annotation framework. Before you can apply CCASH to the annotation task you are interested in, you’ll need to create an Annotation Manager to run on the server, and an Annotation Task to run in your annotators’ browsers. If you create something that you think others might be interested in, please contribute it to the repository!

- Create an Annotation Task by following the tutorial at `https://facwiki.cs.byu.edu/nlp/index.php/Creating_an_Annotation_Task`.
- Create an Annotation Manager by following the tutorial at `https://facwiki.cs.byu.edu/nlp/index.php/Creating_an_Annotation_Manager`.

Do I have to build my application from scratch?

We have already developed some annotation tasks that we are interested in. Feel free to use their pieces as building blocks for your own project!

Example annotation tasks

These are fully formed annotation tasks you can use for reference. See the online documentation for links to demos of these applications and Javadocs for related classes (`https://facwiki.cs.byu.edu/nlp/index.php/CCASH#Example_annotation_tasks`).

- Simple Sentiment Classification (an *extremely* simple task put together for Demo purposes).
- English part of speech tagging - Label sequences of English words with their respective parts of speech from the Penn Treebank Tagset (`http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`).
- Syriac morphological tagging - Label sequences of Syriac words with their respective morphological analyses. This includes separating the prefix and stem from the main word, assigning a grammatical category (Noun, Verb, etc), assigning gender (common, masculine, feminine), and so on.
- Syriac morphological tagging tutorial - The same as normal Syriac morphological tagging, except that after each sentence the annotator receives feedback on how they did and optionally are obliged to try the sentence again.
- Survey - Asks users to answer a series of short answer and multiple choice questions
- User study - Takes an annotator through a predetermined sequence of other tasks.
- Training - Presents annotators with a series of instructions on the left side of the screen while they perform an annotation task on the right side of the screen.

Reusable components

These are reusable components that was have developed while working on our own tasks. Check out the linked javadocs for more information.

- AbstractFileReadingAnnotationManager - reads a list of instances from a file, and (optionally) a list of pre-annotations from another file, and finally records annotations received to a file.

Half-baked Tasks

These are tasks that we have in the incubator.

- Named Entity Tagging - Label noun phrases as Person, Location, Business, etc.

6.4 I have a problem! What should I do?

First consult the CCASH Frequently Asked Questions (`https://facwiki.cs.byu.edu/nlp/index.php/CCASH_Frequently_Asked_Questions`). If your question isn't answered there, send us a note at ccash at cs dot byu dot edu.

Chapter 7

Appendix B: User Study Content

This appendix documents the details of the user study.

7.1 Email invitation

To participants in the Syriac corpus annotation user studies:

We thank you for being willing to participate in user studies leading to the construction of the Syriac Electronic Corpus (`http://cpart.byu.edu/?page=112&sidebar`). The Natural Language Processing (NLP) Lab at Brigham Young University, in association with the Center for the Preservation of Ancient Religious Texts (CPART) and the Oriental Institute at the University of Oxford, is developing cost-efficient methods of annotation in order to make the Corpus's construction feasible. These user studies play an important role in that effort.

You will be participating in highly-focused annotation studies using a tool we call CCASH (Cost-Conscious Annotation Supervised by Humans). The first will be conducted early in 2011. We anticipate that the first study will take between three and five hours to complete. The results from the studies will be used to improve annotation methods and will be key to the future development of important resources to be derived from the Corpus, including an annotated electronic concordance. Participants who are interested may continue to be involved with the project and benefit by using CCASH to annotate their own Syriac texts. Please contact me (Paul Felt, pablofelt at gmail dot com) for more information about the user studies and CCASH, and contact Kristian Heal (kristian_heal at byu dot edu) for more information about the Syriac Electronic Corpus.

To proceed with the first user study, please follow this hyperlink:

`https://facwiki.cs.byu.edu/nlp/index.php/Syriac_User_Study`

Thank you.

Paul Felt, on behalf of the BYU NLP Lab, CPART, and the Oriental Institute, Oxford

7.2 Introductory Webpage

Original location: `https://facwiki.cs.byu.edu/nlp/index.php/Syriac_User_Study`

7.2.1 Introduction to the Project

The desirability of an electronic corpus of Syriac texts has long been recognized (most recently in Lucas Van Rompay's January 2007 Hugoye article). Several localized and limited steps have been made in this direction, most significantly with the Peshitta, and as part of the Comprehensive Aramaic Lexicon project. However, no coordinated and large scale effort has yet been attempted. Since 2001 scholars at the Center for the Preservation of Ancient Religious Texts (CPART) at Brigham Young University (BYU) have been working towards creating a comprehensive electronic corpus of Syriac texts. In 2004 they were joined in this effort by Dr. David G.K. Taylor of the Oriental Institute at the University of Oxford. Working from both printed editions and manuscripts this project aims to systematically acquire accurate electronic copies of all of Syriac literature.

Furthermore, the Syriac Electronic Corpus will include a morphological annotation of each word. Because of the size of the undertaking, some parts of the corpus will be automatically annotated by a machine. More crucial parts of the corpus will be annotated by human annotators.

The Natural Language Processing (NLP) Lab at BYU is developing tools and cost-efficient methods of annotation in order to make the Corpus's construction feasible.

7.2.2 The Value of Corpora

Linguistic annotations of text offer many substantial benefits. Annotations can be used to more reliably find linguistic patterns, explore language usage, track how it changes over time, and discover rare forms. Finally, for a morphologically-rich language like Syriac, annotations can be a practical help to language learners. For example, Semitic language dictionaries tend to list conjugated verb forms after a base form (the dictionary headword). This arrangement is convenient for dictionary users who are already familiar with verb conjugations but can be very frustrating for language learners. Annotations allow language learners to interact more naturally with the text they are learning.

The Syriac morphological annotations we are collecting involve segmenting each word into prefix, stem, and suffix. The stem and suffix are then tagged with morphological information, and the stem is further annotated with the corresponding dictionary headword. The end result will be a morphologically annotated electronic corpus of the Syriac texts: a body of texts where every word is linked to a dictionary entry, and a dictionary where every entry is linked to each of its usages in the corpus.

7.2.3 How Can Machines Help?

Traditionally annotated corpora have been laboriously labeled by hand. Especially for under-resourced languages like Syriac, however, this approach is cost-prohibitive. Research in the field of Natural Language Processing (NLP) and Machine Learning has introduced several possible solutions to this problem. For example, annotation can be cheaper and more accurate when annotators are asked to correct machine predictions rather than annotate from scratch. The BYU NLP Lab has developed a model capable of making such predictions with high accuracy for Syriac. Additionally, it has been shown in some domains that annotation efficiency can be increased by automatically selecting which examples are annotated first, a technique known as Active Learning. The NLP Lab has been involved in improving Active Learning's efficiency and usability in real applications and extending the methodology to better handle unexplored domains such as Syriac morphological tagging.

We have created a web-based annotation tool called CCASH (Cost Conscious Annotation Supervised by Humans) that takes advantage of these methods. As CCASH matures, it will be used to develop the annotated Syriac corpus. In addition, it will be made available to the public for other annotation projects.

7.2.4 The Importance of the User Studies

CCASH and the Syriac Electronic Corpus are both planned as open access resources intended to benefit the field of Syriac studies. The results of the user studies and feedback from users will directly impact the development and the functionality of the annotation tools. These tools will be used to computationally annotate the entirety of the corpus. Moreover, user study participants will also be able to use CCASH to efficiently and completely annotate Syriac texts that they are interested in.

7.2.5 User Study #1

In this user study we are gathering data on the effectiveness of having annotators correct machine predictions rather than annotate from scratch. This technique is called automatic pre-annotation.

As you take the study you will encounter machine predictions of varying quality. Bear in mind that most of these annotations will be of intentionally poor quality, and are not representative of the best our model can do.

We aim to determine how correct machine predictions need to be before they begin to be useful to annotators. This knowledge will help us appropriately apply pre-annotations to difficult texts like poetry.

We are also exploring an enhancement to pre-annotation in which machine predictions are updated in response to an annotator's actions. As you annotate, future pre-annotations will sometimes be updated based on decisions you have already made.

7.2.6 Getting Started

Fonts

The Syriac Computing Center (SyrCOM) of Beth Mardutho provides excellent free Syriac fonts (`http://www.bethmardutho.org/meltho/`) that are compatible with the Windows Operating System.

If you are running an up-to-date browser, your browser should automatically load and use these fonts to display Syriac text in the Serto script. If you encounter any font-related problems, however, try downloading and installing the fonts manually. If Syriac text still doesn't render correctly, try using a different browser or setting your browser font manually (`http://kb.iu.edu/data/aojz.html`).

Unfortunately, the Beth Mardutho fonts are not entirely compatible with Mac OS X or Linux. For this user study, we recommend using **Windows XP, Vista, or 7**.

Compatible Browsers

Please use a browser that is compatible with CCASH. The most recent versions of the following browsers work well with CCASH:

- Firefox
- Chrome
- Safari

The following browsers do NOT work with CCASH:

- Internet Explorer
- Opera

Create an account with CCASH

Navigate to `http://cash.cs.byu.edu/Ccash` and click the button that says "Register". When you are done registering, you will be sent a verification email message. Open that email message using your favorite mail reader and click on the link to activate your account. Then you will be ready to annotate.

Start Annotating

Navigate to `http://cash.cs.byu.edu/Ccash` and log in with your newly created username and password. You will see a list of all the projects you have been assigned to annotate. For now, that list contains only the user study. Click the button that says "Annotate".

Resources

Resources you may want to use while annotating include:

- A reference summary of the training you will receive inside CCASH (`http://nlp.cs.byu.edu/public/AnnotatorTraining.pdf`)
- A Compendius Syriac Dictionary by Robert Payne Smith (`http://www.tyndalearchive.com/TABS/PayneSmith/index.htm`)
- Compendious Syriac Grammar by Theodor Nöldeke (`http://cpart.byu.edu/files/N%C3%B6ldeke_Compendious%20Syriac%20grammar.1904.pdf`)
- Syriac Verb Tables by David Taylor (`http://nlp.cs.byu.edu/public/Syriac%20Verb%20paradigms%20(DGKT%201.1).pdf`)

7.3 In-study Roadmap

Thank you for participating in the Syriac Ccash User Study. During this study you will be asked to annotate text with morphological information. The purpose of this study is two-fold. We are measuring both annotator speed and accuracy; therefore, please remove all distractions so that you can complete the task quickly and accurately. As you annotate, some of the sentences you encounter will include suggested annotations provided by computational models having varying levels of accuracy.

Tutorial and Resources

You will first work through a tutorial to familiarize yourself with the software and the annotation task. On the wiki page you were given access to a reference summary of the tutorial, as well as other resources you may wish to use. For your convenience, here is the `link:https://facwiki.cs.byu.edu/nlp/index.php/Syriac_User_Study#Resources`

Practice Sentences

After the tutorial, you'll be presented with four practice sentences to annotate. Some of the practice sentences will have suggestions as discussed above and others will have no suggestions. After you have completed each example, you will be shown the correct tags in order to let you see where your tags differ from ours. On the final practice sentence, you will be expected achieve a high level of accuracy before continuing.

The Main Study

After completing the practice sentences, you will be ready to tag the main study. During this time, we will monitor both your speed and your accuracy, so we ask that you remove all distractions and do your best work! When you are done, you will take an exit survey and see a message indicating that you are finished. Should you wish to continue, however, you will have the option of annotating some additional sentences.

7.4 Data Usage Agreement

By contributing to this site you give the BYU Natural Language Processing Lab and the BYU Center for the Preservation of Ancient Religious Texts (CPART) unrestricted ownership of and rights to use the data that you generate for this user study in any way that we deem reasonable. Our plans include a detailed analysis of the data, publication in papers about efficient annotation techniques, published corpora and concordances, and better interactive annotation models.

At the same time, when we share or publish the data that you generate, we will ensure that you are not personally associated with any of the data.

Press "Continue" to accept these terms and proceed.

7.5 Entrance Survey

- What is your primary language?
  - English
  - Arabic
  - Italian
  - Hebrew
  - Turoyo
  - Turkish
  - Dutch
  - Swedish
  - German
  - Russian
  - Flemish
  - French
  - Other
- How much formal linguistic training do you have?
  1. None
  2. A little

    3. About one college class
    4. Several college classes
    5. One or more college level degrees in linguistics or a related field

- How much formal Syriac language instruction have you received?
    - None
    - A little
    - About one college class
    - Several college classes
    - One or more college level degrees in Syriac or a related field
- Have you ever taught the Syriac language?
- If so, in what setting?
- Have you ever conducted research on original Syriac texts?
- How proficient are you at Syriac morphological tagging?
    1. Poor
    2.
    3. Average
    4.
    5. Excellent
- How comfortable do you feel using computers?
    1. Uncomfortable
    2.
    3. Average
    4.

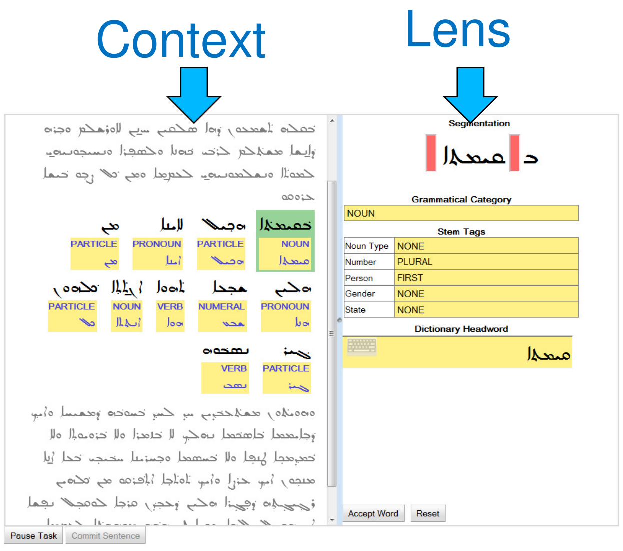
5. Very comfortable

- Where are you physically located? (country, city)

7.6 Annotator Training

Syriac Morphological Annotation with CCASH

- To the right of these instructions, you see a left panel and a right panel. The right panel is the annotation editor or "lens." The left panel is the text to be annotated or "context". The word that you are currently annotating appears enlarged at the top of the lens.
- The lens contains five total sub-tasks to complete for each word:
  - Segmentation
  - Grammatical Category
  - Stem Tagging
  - Suffix Tagging
  - Identifying a Dictionary Headword



Segmentation

Grammatical Category

NOUN

Stem Tags

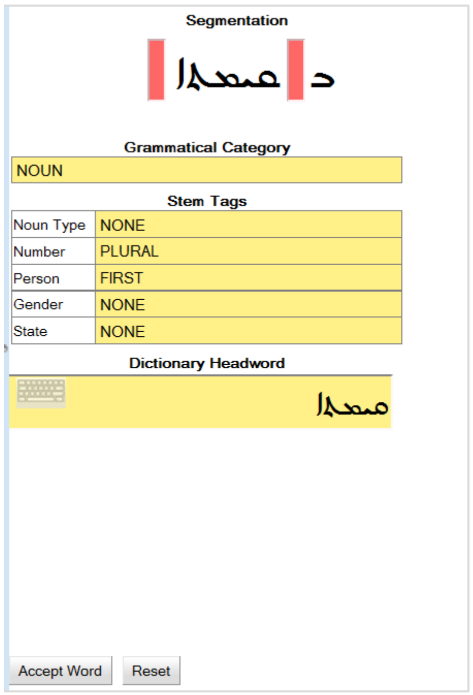
Noun Type	NONE
Number	PLURAL
Person	FIRST
Gender	NONE
State	NONE

Dictionary Headword

Accept Word Reset

Segmentation

- Your first task is to edit the segmentation of the word. The current segmentation is indicated by the two red bars in the word.
- The prefix is the text that appears to the right of the rightmost bar. The suffix is the text to the left of the leftmost bar, and the stem is the text in between. Prefix and suffix clusters are segmented together.
- A note about browsers and focus: Most browsers indicate what element is currently being operated on by outlining it with a rectangle or highlighting it in some way. If your browser is not focused on the segmentation task, then try clicking on the segmentation task to focus it.
- Try changing the segmentation in two ways:
  - <u>Keyboard</u>: while the segmentation widget is in focus, use the left and right **arrow keys** to change the suffix marker. Now hold down **shift** and use the arrow keys to change the prefix marker.
  - <u>Mouse</u>: **left click** to set the suffix position, and **right click** to set the prefix position.
- Note that suffix tags become available or unavailable depending on whether or not you identify a suffix segment.



Segmentation

Grammatical Category

NOUN

Stem Tags

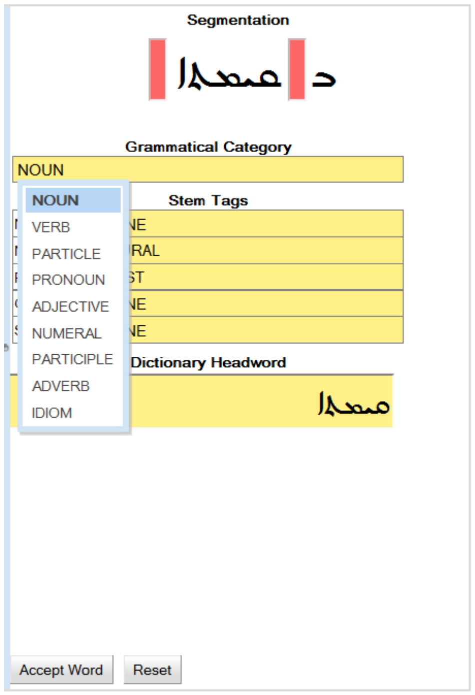
Noun Type	NONE
Number	PLURAL
Person	FIRST
Gender	NONE
State	NONE

Dictionary Headword

Accept Word Reset

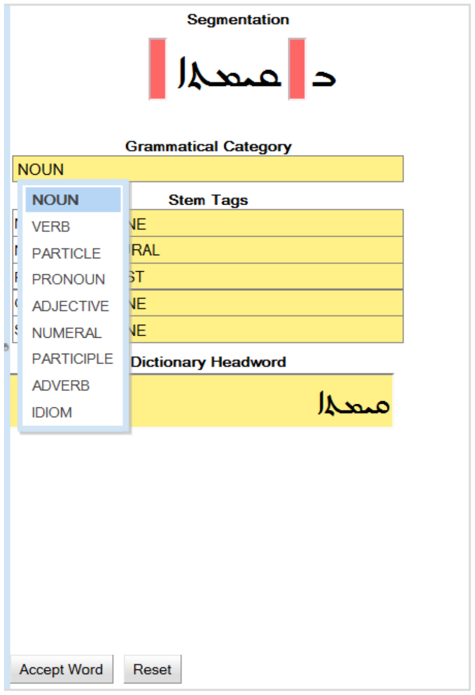
Grammatical Category

- Your second task is to select or correct the grammatical category of the word.
- Try changing the grammatical category in two ways:
    - Keyboard: press **tab** to move from the Segmentation task down to the Grammatical Category task. A list of possible values will appear. **Begin typing** the name of the value you wish to select, and when the correct value is selected, press **tab again** to move on to the next field.
    - Mouse: click the Grammatical Category field. A list of possible values will appear. Click the name of the value you wish to select, and notice that the next field is automatically selected.
- Note that each grammatical category has its own set of stem tags, i.e., additional properties associated with the grammatical category. They become available or unavailable depending on your choice of grammatical category..



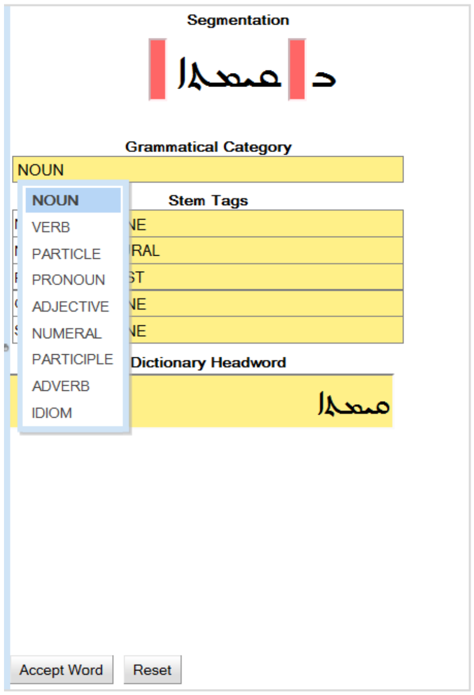
Notes on Grammatical Categories

- As a general note, annotation corresponds to the visible form rather than the function of the token being analyzed. This impacts participles in particular, which may function adjectivally, or be substantivized. In either of these cases the token should still be annotated as a verb.
- Particle
    - Includes prepositions, interjections, conjunctions and adverbial particles. If in doubt, refer to the list provided in the training materials.
- Noun
    - Includes proper nouns and common nouns. Substantivized participles should be annotated under verb and participle.
- Pronoun
    - Includes personal, demonstrative and interrogative pronouns.
- Adjective
    - Refers to adjectives proper; participle adjectives should be annotated under verb and participle.



Notes on Grammatical Categories

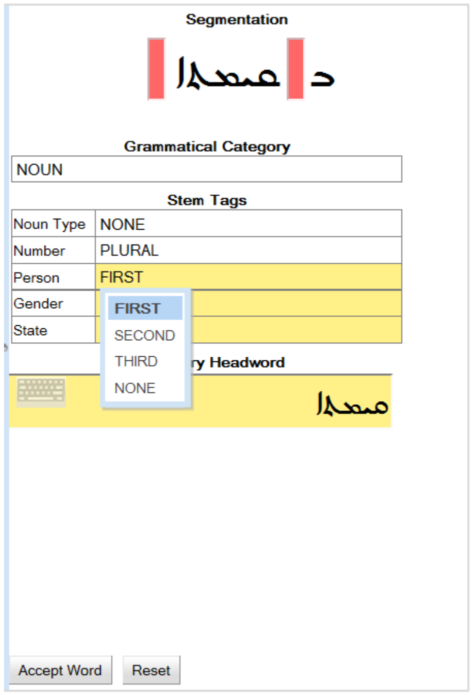
- Adjective
  - Refers to adjectives proper; participle adjectives should be annotated under verb and participle.
- Verb
  - Includes both regular and demoninative verbs in all their conjugations.
- Adverb
  - Includes all formal adverbs, such as adverbs of quality formed with the termination –aith. Adverbial particles should be annotated as particles.
- Numeral
  - Includes cardinals, Ordinals, and ciphers.
- Idiom
  - Includes only compound forms


Segmentation

Grammatical Category: NOUN

Stem Tagging

- Your third task is to describe the properties of the stem by choosing values for applicable stem tags.
- Try changing the stem tags in two ways:
  - Keyboard: Use the keyboard in the same manner as when selecting the Grammatical Category. To accept the current field and move forward, press **tab**. To accept the current field and move backward, hold down **shift** and press **tab**.
  - Mouse: Use the mouse in the same manner as when selecting the Grammatical Category.


Segmentation

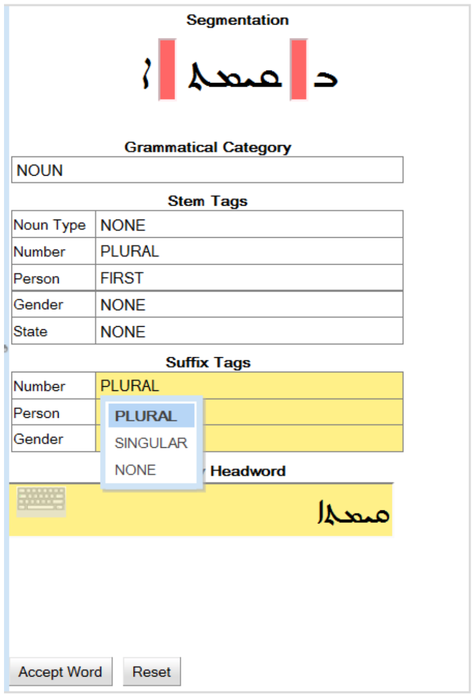
Grammatical Category: NOUN

Stem Tags

Noun Type	NONE
Number	PLURAL
Person	FIRST
Gender	
State	

Suffix Tagging

- Your fourth task is to describe the properties of the suffix by choosing values for applicable suffix tags.
- Try changing the suffix tags in the same two ways you changed the stem tags.



Segmentation

ܐ̈ | ܡܫܡܫ | ܕ

Grammatical Category

NOUN

Stem Tags

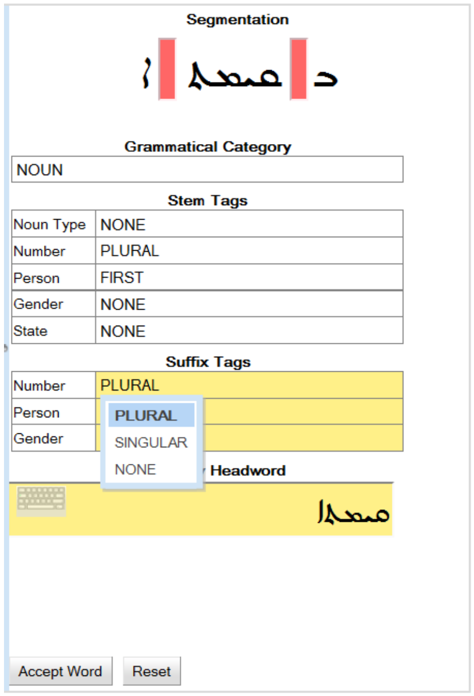
Noun Type	NONE
Number	PLURAL
Person	FIRST
Gender	NONE
State	NONE

Suffix Tags

Number	PLURAL
Person	
Gender	

Notes on Suffix Tags

- Suffix Contraction
  - Enclitic personal pronouns that have coalesced with participles should be segmented and annotated as a suffix. In this case, however, the user is given the chance to indicate whether this is a “suffix contraction” rather than an object suffix.



Segmentation

ܐ̈ | ܡܫܡܫ | ܕ

Grammatical Category

NOUN

Stem Tags

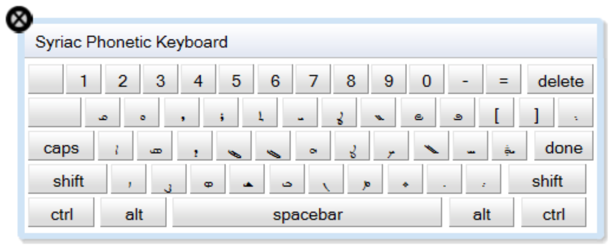
Noun Type	NONE
Number	PLURAL
Person	FIRST
Gender	NONE
State	NONE

Suffix Tags

Number	PLURAL
Person	
Gender	

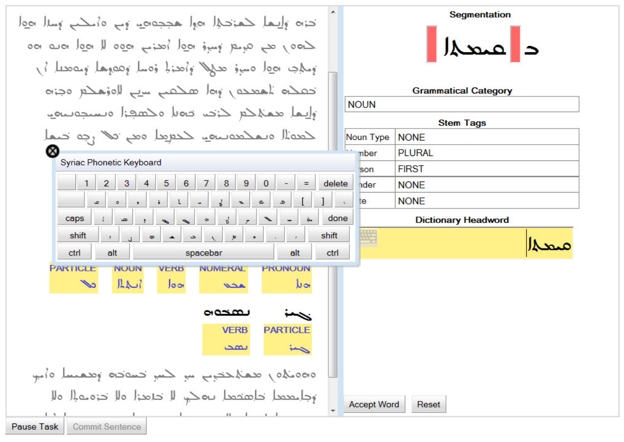
Dictionary Headword

- Your fifth task is to identify the dictionary headword of the stem.
- This field requires that you type in Syriac. If your machine is not already configured for typing Syriac characters, you may wish to use the Virtual Keyboard provided.
- Try viewing the Virtual Keyboard in two ways:
  - Keyboard: while your cursor is in the Dictionary Headword field, press **escape**
  - Mouse: click the keyboard icon in the Dictionary Headword field
- Now hide the Virtual Keyboard in two ways:
  - Keyboard: while your cursor is in the Dictionary Headword field, press **escape**
  - Mouse: click the X at the top left corner of the Virtual Keyboard
- Now try using the Virtual Keyboard in two ways:
  - Keyboard: With your cursor inside the dictionary headword textbox, **press keys** on your keyboard.
  - Mouse: click the keys on the virtual keyboard with your mouse



Notes on Dictionary Headwords

- A dictionary headword is the uninflected form of the stem. E.g., for nouns this is the emphatic form. For verbs this is the uninflected third masculine singular Peal perfect form that is usually found as the headword in Payne-Smith's dictionary..


Segmentation

Grammatical Category

NOUN

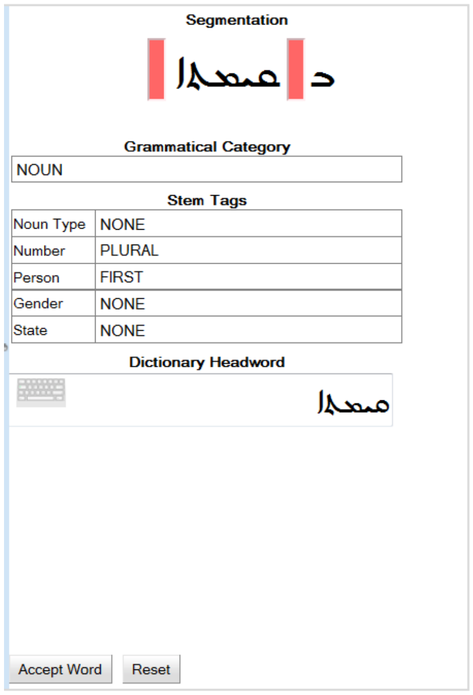
Stem Tags

Noun Type	NONE
mber	PLURAL
son	FIRST
nder	NONE
te	NONE

Dictionary Headword

Reset

- While annotating a word, you may wish to reset the annotation to its original values. Do this by selecting a word you have annotated and pressing the “Reset” button.



Segmentation

ܡܥܡܕܐ ܕ

Grammatical Category

NOUN

Stem Tags

Noun Type	NONE
Number	PLURAL
Person	FIRST
Gender	NONE
State	NONE

Dictionary Headword

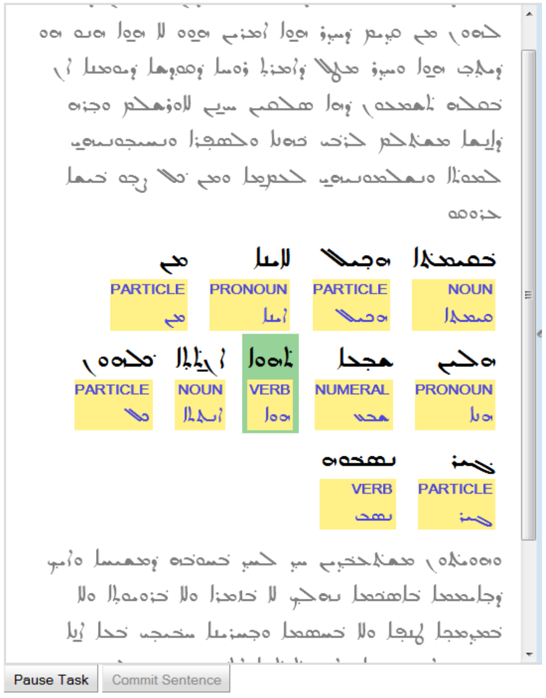
ܡܥܡܕܐ

Intentionally poor predictions

- You will notice during the study that most of the automatically-generated annotations you encounter are of poor quality. Please bear in mind that this is necessary for the purposes of this study, and that this is not representative of the best our models can do!

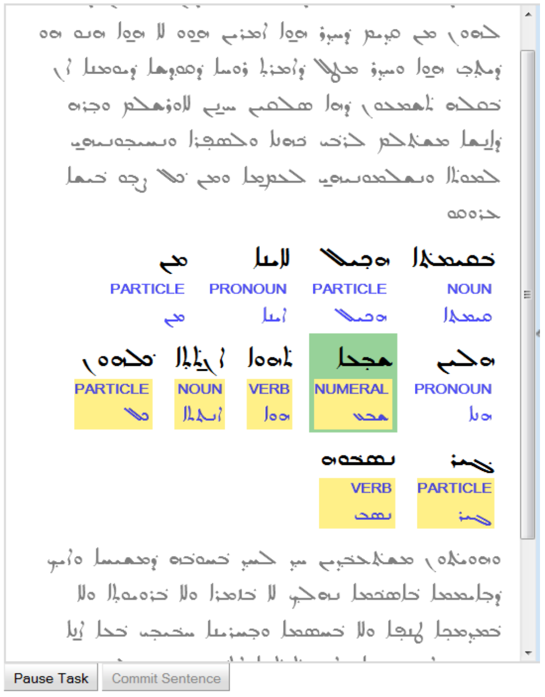
Navigation within a Sentence

- Until now, these instructions have been neglecting an important part of the interface. The left portion of the screen (the "context") shows the sentence you are currently working on and your position in it (indicated by a word with a green background). Also, some context before and after the current sentence is provided. The additional context is grayed out to indicate that it is not to be annotated.
- Although it may be most efficient to work through a sentence sequentially from beginning to end, you may annotate in any order you wish.
- Try navigating through a sentence in two ways:
  - <u>Keyboard</u>: hold down **control** and press the left and right **arrow keys**
  - <u>Mouse</u>: click the word you wish to annotate
- Notice that below each word in the sentence is the grammatical category currently selected for that word.



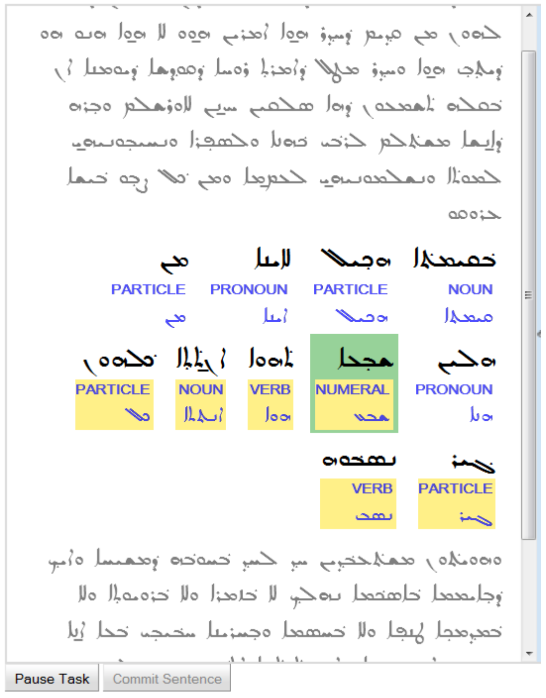
Accepting Words

- You will have noticed that when you change the value of a field in the lens, its background color changes from yellow to white. The white background signals that the field in question has been accepted by you. When all fields within a word are white, a word is considered complete. If at any time you wish to accept an entire word without manually touching every field, you may do so by accepting the word using the "Accept Word" button.
- Note that when a word is accepted (all of its fields are white), then the background of its grammatical category in the sentence on the left portion of the screen changes from yellow to white. This lets you know which words in the sentence still need your attention.
- Try accepting a word in two ways:
  - <u>Keyboard</u>: press **control** + **enter**
  - <u>Mouse</u>: click the “Accept Word” button



Committing Sentences

- After you have accepted every word in the sentence, the "Commit Sentence” button will become active.
- After accepting all the words in a sentence, try committing a sentence in two ways:
  - Keyboard: press **control** + **alt** + **enter**
  - Mouse: click the “Commit Sentence” button
- Note that committing a sentence is permanent. Once you have committed, you will not be able to revisit your decisions for any of the words in the sentence.



7.7 Practice Sentences

You are about to work through four practice sentences. The purpose of this exercise is to give you experience using the software you've just learned, and to give you time to familiarize yourself with the tags and the resources that you will be working with. After you commit the practice sentences, the correct tagging for each word will be shown along with the tagging that you selected.

For the last practice sentence you will be required to achieve a certain standard of correctness before moving on. All segmentation and grammatical category choices must be correct, but you will be allowed to make other mistakes on up to two words.

It you disagree with the reference tagging, please consult your reference materials to understand why a particular value applies. One of the purposes of these practice sentences is to expose you to concrete examples of the tags applied to the data. We understand that you will likely disagree with some of our tagging decisions. However, for consistency's sake we ask that during this user study you conform as best you can to the tagging conventions you encounter in these practice sentences and the provided reference materials.

Practice Sentence #1: Annotating Without Suggestions

For the following sentence, you will annotate each word without assistance. For tips on entering tags with your mouse or keyboard, consult your tutorial review.

Practice Sentence #2: Annotating With Suggestions

For the following sentences, you will notice that the words are pre-labeled with machine-generated guesses. It is important to note that sometimes these predictions will be updated as you annotate. These updates appear to the side of their respective fields as blue hyperlinks which you may accept by clicking.

Practice Sentence #3: No Instructions

Practice Sentence #4: Raising the Bar

For this sentence, you will be required to achieve a certain standard of correctness before moving on. All segmentation and grammatical category choices must be correct, but you will be allowed to make other mistakes on up to two words.

7.8 Main Study

During the main study you will see sentences pre-labeled with various degrees of accuracy. Remember that most of the pre-labels you will encounter will be of intentionally poor quality, and are not representative of the best our model can do!

Note that in the main study you will be annotating clauses and not necessarily complete sentences; however, you should have access to the information you need in the surrounding context.

During this time we ask you to please give this study your undivided attention. If you do need to pause during the study for any reason, please press the "Pause" button so that we can accurately track the time required to complete the task. If you need to leave for long periods of time, you may also log out. Bear in mind, however, that when you log out, changes to the sentence you are currently working on will be lost. For this reason try to log out only between sentences. In addition, avoid using time when you are paused or logged out to work on the study (e.g., examine reference materials). We estimate that the average participant will need between two and five hours to complete the main portion of the study.

Before you begin the study please turn off any electronic devices, close your email, remove any other distractions that you can, and maximize your browser. If you are using Firefox or Internet Explorer and Windows, you can put your browser in fullscreen mode by pressing F11. Once you are completely ready, press the button to continue.

User Study #1 Clauses

1: ܡܛܠܗܘܢ ܕܟܠܒ ܗܐ ܡܟܝܠ ܘܡܚܙܝܐ .

2: ܘܢܦܩ ܠܘܬ ܬܝܒܐ ܩܪܝܬܐ ܕܡܠܟܘܬ ܠܟܦܪܐ ܘܠܡܕܝܢܬܐ ܘܠܡܠܟܘܬܐ .

3: ܘܐܬܐ ܥܡܗ ܒܥܠܗ ܠܫܡܝܐ ܕܒܘܪܬܐ .

4: ܘܒܪܘܟܬܗ ܡܠܟܐ .

5: ܘܡܫܬܘܬܐ ܚܡܪܐ .

6: ܢܦܩܝܢ ܗܘܘ ܓܝܪ ܕܢܩܘܡ ܠܗ ܡܕܡ ܡܠܟܐ .

7: ܐܝܟܐ ܥܒܕܬ ܐܝܢ ܠܗ .

8: ܘܐܬܟܢܫܘ ܠܘܬ ܐܢܫܐ ܘܐܡܪܘ ܗܘܘ ܕܢܗܘܐ ܩܕܡܘܗܝ ܗܘܐ ܀

9: ܕܐܢܫܐ ܡܠܬܐ ܢܫܡܥܘܢ ܠܒܢܘܗܝ ܘܠܚܒܪܐ .

10: ܐܡܪ ܠܗ ܡܠܟܐ .

11: ܡܫܡܥܝܢ ܕܫܡܥܢ ܙܒܢܬܐ ܡܢ ܛܠܝܘ .

12: ܡܢܘ ܡܫܬܒܩ ܠܠܒܐ ܩܕܝܫܐ ܡܢ ܠܚܡܐ ܕܒܪܐ .

13: ܕܡܢ ܗܝܐ ܐܘܟܝܬ ܠܐ ܦܪܘܫ .

14: ܐܝܟ ܗܘܐ ܠܗ ܓܝܪ ܠܡܠܟܘ .

15: ܡܚܝܐ ܕܦܓܪܝܢ .

16: ܕܘ ܓܝܪ ܢܘܚ ܢܫܡܘ ܚܠܦܝܢ .

17: ܡܦܪܫ ܐܬܪܐ ܠܠܟܝ ܘܐܬܠܗܘܢ ܠܗ .

18: ܗܢܐ ܐܘܡܢܘܬܐ ܝܕܥ ܐܢܬ ܕܐܚܪܢ .

19: ܩܕܝܫܐ ܓܝܪ ܗܘܢܘ ܡܠܟܝ .

20: ܗܢܐ ܐܡܪ ܠܐ ܝܕܥ ܐܢܐ .

21: ܘܢܫܡܥܘܢ ܡܢ ܕܩܕܡ ܕܩܕܘܫܠܘܢ .

22: ܘܠܐ ܡܢܫܡܥ ܘܢܘܢ .

23: ܚܡܪܐ ܕܚܕܘܬܐ ܠܗ ܠܐ ܡܬܚܙܝܐ .

24: ܘܕܠܐ ܕܝܠܗ ܡܢܗ ܐܘܪܚ .

25: ܘܦܪܫ ܗܘ ܡܢ ܡܠܗܘܢ ܐܦܫܬܐ .

26: ܐܠܐ ܗܢܐ ܗܘܠܐ ܙܒܢܐ ܡܫܬܠܡ ܡܓܢܘܢܐ .

27: ܘܐܡܪ ܐܦܠܐ ܡܕܡ ܕܗܘ ܥܡ ܩܠܝܠܐ ܣܓܝ .

28: ܘܐܝܟ ܠܐܪܥܗܘܢ ܠܡܠܟܗ ܕܢܫܬܪ ܙܒ ܚܝܠܐ .

29: ܘܡܠܗܘܢ ܐܝܟܢ ܕܢܚܫܘܢ ܗܘܘ ܠܐܦ ܕܡܫܡܥ ܗܘܘ .

30: ܘܡܢ ܠܐܪܥܗ ܢܘܚ ܠܥܠܡ ܐܝܟܐ ܕܐܒܝܕܗ ܡܠܟܗ .

7.9 Exit Survey

The following exit survey concludes the required portion of the user study. If you have feedback related to a particular item in the user study, a reference sheet of the clauses you tagged can be found at `http://nlp.cs.byu.edu/public/userstudy1-clauses.pdf`

- How accurately do you think you performed on this experiment?

    1. Poorly
    2.
    3. Average
    4.
    5. Excellent

- Did you have the reference sheet by your computer while you did the study?
- Did you pause the program as necessary during the annotation process to ensure accurate timing?
- Approximately how much time did you spend away from the computer without pausing?
- Overall, how useful did you find the preannotations offered to you?

    1. They hurt my performance
    2.
    3. They neither helped nor hurt
    4.
    5. They helped

- Did the order of the tasks feel natural? (segmentation, grammatical category, stem tagging, suffix tagging, dictionary headword)
- If not, what do you think would have been a better order?

- We are trying to improve this application in preparation for a large-scale annotation effort. Please offer any comments or suggestions that occurred to you as you used it.
- Did the application fit comfortably in your browser?
- Did you use any additional reference materials to complete the study?
- If so, what additional reference materials did you use to complete the study?

Chapter 8

Appendix C: Sentence level analysis

Before the word-level analysis described in Chapter 4, we conducted a sentence-level analysis of the user study annotation times. This analysis was later adapted to a word-level analysis so as to be more comparable to our statistical significance tests. The significance tests must be performed on word-level data because they do not account for varying sentence lengths.

8.1 Introduction

Supervised machine learning techniques are designed to learn patterns from hand-labeled data, and are more effective with large amounts of data.

The natural language processing (NLP) lab at BYU is working with the Center for the Preservation of Ancient Religious Texts (CPART) at the Maxwell Institute to label an extremely large corpus of texts written in Classical Syriac, an historically important dialect of Aramaic (`http://cpart.byu.edu/?page=112&sidebar`). Each Syriac word is divided into prefix, suffix, and stem, and the stem and suffix are subsequently labeled with detailed grammatical information.

Labeling the Syriac New Testament took about 15 years. CPART wants to annotate at least 100 times more data than the New Testament. To accomplish this they must label more efficiently.

One way of potentially making labeling more efficient is **pre-labeling** (PL): having a machine learned model provide a candidate label for each word. Annotators thus need only review candidate labelings and correct mistakes rather than create each label from scratch. A potential way of improving pre-labeling is to listen for annotator corrections and update candidate labels appearing later on in the word or sentence based on the annotator's corrections. We call this **correction propagation** (CP).

It seems likely that pre-labels provided by a sufficiently high quality model will be helpful in reducing annotation time. On the other hand, it is possible that sufficiently poor pre-labels would actually hinder annotators. Because the corpus to be labeled is heterogeneous, the model will perform worse on some parts of the corpus than others. Thus it would be desirable to know what level of model performance is required before pre-labeling becomes helpful rather than harmful.

Data/Model

We ran a user study in which 9 annotators sequentially labeled a set of 30 sentences chosen at random from the Acts of Judas Thomas, a text they had not before encountered. As annotators began each sentence, they were randomly assigned to annotate under 1 of 16 conditions corresponding to a particular degree of automatic assistance. Under the first condition, annotators were given no automatic assistance at all, and annotated each word from scratch. Under the second condition, annotators were given perfect pre-labels generated by two experts annotators. Under the remaining 14 conditions, annotators were given pre-labels with and without correction propagation generated by a supervised model trained on the already-labeled New Testament to one of 7 accuracy levels (measured against a gold standard generated by two expert annotators). The conditions can be summarized as $\{\text{no PL, perfect PL}\} + (\{25, 36, 47, 58, 68, 79, 90\} \times \{\text{with CP, without CP}\})$. The time that each annotator time took to annotate each sentence was measured, resulting in 270 measurements; between 14 and 16 per condition.

The number of seconds taken to annotate a sentence was modeled in 4 different ways. Each model treats sentence time as a normally distributed sum, and each models adds one more term into that sum than the previous model. The first model $\boldsymbol{\alpha\beta}$, models the contribution of the sentence's position in the study, β_l, as well as the contribution of the condition under which the sentence was annotated, α_k. The second model, $\boldsymbol{\alpha\beta\delta}$, adds the contribution of the length of each sentence, δ_s. The third model $\boldsymbol{\alpha\beta\delta\theta}$ adds the contribution of the the annotator who annotated the sentence, θ_i, and the fourth model, $\boldsymbol{\alpha\beta\delta\theta\gamma}$, adds the contribution of the condition under which the previous sentence was annotated, γ_j. For brevity's sake, I explain only the priors, likelihood, and log posterior numerator $ln(g)$ of the final, most complex model, since the previous three can be derived from that by simply removing the relevant portions of the model, and complete conditionals derived from $ln(g)$

Likelihood

$f^*(y_{ijkls}|\theta_i, \gamma_j, \alpha_k, \beta_l, \delta_s)$
$\sim N(\delta_s + \theta_i + \gamma_j + \alpha_k + \beta_l, \sigma^2)$

$L(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}) =$
$\prod_{y_{ijkls} \in \mathbf{y}} f^*(y_{ijkls}|\theta_i, \gamma_j, \alpha_k, \beta_l, \delta_s) =$
$(2\pi)^{-\frac{N}{2}}(\sigma^2)^{-\frac{N}{2}} e^{-\frac{1}{2\sigma^2}\sum_{y_{ijkls}\in\mathbf{y}}(y_{ijkls}-\delta_s-\theta_i-\gamma_j-\alpha_k-\beta_l)^2}$

Priors

Unfortunately, space constraints prohibit justifying my priors. Justifications are outlined in a separate document, available upon demand. Contributions whose existence and quantity that are in question are distributed normally with a high variance around 0, so as to allow them to be learned from the data.

$\sigma^2 \sim Gamma(2, 15)$
$\delta_s \sim N(m * len_s + b, \sigma^2_\delta)$
$m \sim Gamma(35, 2)$
$b \sim Gamma(9, 1.5)$
$\sigma^2_\delta \sim Gamma(1.5, 10)$
$\theta_i \sim N(\mu_\theta, \sigma^2_\theta)$
$\mu_\theta \sim N(0, 400)$
$\sigma^2_\theta \sim Gamma(12, 3)$

$\alpha_k \sim N(0, 400)$

$\gamma_j \sim N(0, 400)$

$\beta_l \sim N(0, 400)$

ln(g)

$$ln(\Pi(\Theta|\mathbf{y})) \propto ln(g(\Theta|\mathbf{y})) =$$

$$-\tfrac{N}{2}ln(\sigma^2) - \tfrac{1}{2\sigma^2}\textstyle\sum_{y_{ijkls}\in\mathbf{y}}(y_{ijkls} - \delta_s - \theta_i - \gamma_j - \alpha_k - \beta_l)^2 + ln(\sigma^2) - \tfrac{\sigma^2}{15} - \tfrac{\mu_\theta^2}{800} + (11 - \tfrac{A}{2})ln(\sigma_\theta^2) - \tfrac{\sigma_\theta^2}{3} - \textstyle\sum_{i=1}^{A}\tfrac{(\theta_i-\mu_\theta)^2}{2\sigma_\theta^2} - \textstyle\sum_{j=1}^{K}\tfrac{\gamma_j^2}{800} - \textstyle\sum_{k=1}^{K}\tfrac{\alpha_k^2}{800} - \textstyle\sum_{l=1}^{L}\tfrac{\beta_l^2}{800} + 34ln(m) - \tfrac{m}{2} + 8ln(b) - \tfrac{b}{1.5} + 0.5ln(\sigma_\delta^2) - \tfrac{\sigma_\delta^2}{10} - \tfrac{S}{2}ln(2\pi\sigma_\delta^2) - \textstyle\sum_{s=1}^{S}\tfrac{(\delta_s - m*len_s - b)^2}{2\sigma_\delta^2}$$

8.2 Results

	-PL	25	36	47	58	68	79	90	*PL
+CP	NA	-0.9	32.9	21.8	0.1	-15.1	-1.7	8.9	NA
-CP	8.9	27.2	11.4	35.9	29.1	-11.2	**-30.3**	**-23.4**	**-65.6**

Table 8.1: Mean contribution of each annotation condition (α_k). Conditions whose values are different from α_0 with greater than 95% probability are bolded.

I obtained 500,000 samples from each of the models. For all models, all parameter acceptance ratios were between .2 and .4, and the trace plots didn't show excessive wandering or sticking behavior. After thinning by 100, all posteriors passed the Raftery-Lewis Diagnostic with IRLs less than 5. Additionally, most posteriors also had Geweke z-scores well under 1.96. However, the Geweke scores appeared to be sensitive to random runs. Some posterior z-scores bounced back and forth across the 1.96 threshold on different runs. However, since the acceptance ratios, trace plots, and IRLs look okay I think these posteriors are acceptable.

I compared the four models using both their Bayes Factors and Deviance Information Criterion in the following table.

The most complex model was favored by both metrics, so I use it in order to draw conclusions. These numbers were quite stable across random runs, changing only in the tenths decimal

	$\boldsymbol{\alpha\beta}$	$\boldsymbol{\alpha\beta\delta}$	$\boldsymbol{\alpha\beta\delta\theta}$	$\boldsymbol{\alpha\beta\delta\theta\gamma}$
Bayes Factor	-3000.3	-2378.1	-2361.7	-2358.2
DIC	4328.5	4015.6	3870.6	3867.1

Table 8.2: Model Comparison

place. I also compared the models using the Bayes Chi-squared goodness of fit test, but all the models performed equally badly according to that metric.

The $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, $\boldsymbol{\delta}$, and $\boldsymbol{\gamma}$ variables were mainly nuisance parameters to account for effects that might confuse our analysis of $\boldsymbol{\alpha}$, the effect of the condition under which a sentence was annotated. However, γ showed an interesting trend, although it was not significant at the 90% probability level. It appeared that having a previous sentence with low quality PL tended to increase the time required to annotate the subsequent sentence, perhaps because the annotator learned to second-guess the model. Similarly, having a previous sentence with high quality PL tended to decrease the time required to annotate the subsequent sentence, perhaps because the annotator had learned to trust the model. Table 8.1 lists the mean time contribution of each α_k.

8.3 Conclusions

Pre-labels

The bottom row of Table 8.1 shows a clear trend. According to this data and model, pre-labels below about 70% accuracy appear to reduce annotator speed whereas those above increase annotator speed. This is encouraging since it requires relatively little data to train an automatic annotator to the 70% accuracy level. Also, considering that the average unassisted annotation time in the study was 250 seconds, the 20 to 30 second gain provided by high quality pre-labels represents a 10% increase in speed.

Correction Propagation

The top row of Table 8.1 shows the mean contribution of using both pre-labels and correction propagation is quite noisy. Calculating the probability $p(\alpha_{Acc=a,CP=1} < \alpha_{Acc=a,CP=0})$ shows that correction propagation helps beyond simple pre-labeling with probability greater than 0.9 for 25% and 58% accurate pre-labels. Similarly, calculating the probability that the presence of correction propagation hurts compared with simple pre-labeling at each accuracy level, $p(\alpha_{Acc=a,CP=1} > \alpha_{Acc=a,CP=0})$, shows that correction propagation hurts with probability greater than 0.9 for 36%, 79%, and 90% accurate pre-labels.

These results are relatively weak (p <.95), and they are also rather noisy. However, it appears that for the higher quality pre-labels, it may be better not to use correction propagation. One possible explanation for this behavior is that seeing a pre-labels change might break an annotator's train of thought while she considers the new pre-label.

Chapter 9

Appendix D: Derivation of Complete Conditionals for the Bayesian Analysis

9.1 Introduction

This appendix walks through the process involved in calculating our unnormalized joint posterior distribution, $ln(g)$, and then using that as a basis for finding the complete conditionals of each parameter. We model the number of seconds taken to annotate a word y_{hatbro} as a combination of the following variables:

9.2 Variables

- σ^2 Variance common to all words
- θ_h Annotators
- α_a Current condition
- τ_t Grammatical Category
- β_b Bucketed word position (0,1,2,3+)
- ρ_r Hyperlinks clicked
- ω_o Hyperlinks shown
- $kappa$ Offset common to all words

9.3 Priors

Prior justifications may be found in the main paper. Gamma distributions are parameterized by shape and scale.

- $\sigma^2 \sim Gamma(50, 50)$
- $\theta_h \sim N(0, \frac{40}{3})$
- $\alpha_a \sim N(0, \frac{40}{3}) =$
- $\tau_t \sim N(0, \frac{40}{3})$
- $\beta_b \sim N(0, \frac{40}{3})$
- $\rho_r \sim N(0, \frac{40}{3})$
- $\omega_o \sim N(0, \frac{40}{3})$
- $kappa \sim N(90, \frac{50}{3})$

9.4 Likelihood

The density of a single data point is distributed as

$y_{hatbro}|\theta_h, \alpha_a, \tau_t, \beta_b, \rho_r, \omega_o, \kappa \sim N(\theta_h + \alpha_a + \tau_t + \beta_b + \rho_r + \omega_o + \kappa, \sigma^2)$

Assuming that the probability of each data point is independent, the likelihood, or probability of the data set, may be written as the produce of the probability of each data point.

$L(\mathbf{y}|\Theta) = L(\mathbf{y}|\sigma^2, \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{\beta}, \boldsymbol{\rho}, \boldsymbol{\omega}, \kappa)$

$= \prod_{y_{hatbro} \in \mathbf{y}} p(y_{hatbro}|\sigma^2, \theta_h, \alpha_a, \tau_t, \beta_b, \rho_r, \omega_o, \kappa)$

$= \prod_{y_{hatbro} \in \mathbf{y}} (2\pi)^{-\frac{1}{2}} (\sigma^2)^{-\frac{1}{2}} e^{-\frac{1}{2\sigma^2}(y_{hatbro} - (\theta_h + \alpha_a + \tau_t + \beta_b + \rho_r + \omega_o + \kappa))^2}$

$= \prod_{y_{hatbro} \in \mathbf{y}} (2\pi)^{-\frac{1}{2}} (\sigma^2)^{-\frac{1}{2}} e^{-\frac{1}{2\sigma^2}(y_{hatbro} - \theta_h - \alpha_a - \tau_t - \beta_b - \rho_r - \omega_o - \kappa)^2}$

$= (2\pi)^{-\frac{N}{2}} (\sigma^2)^{-\frac{N}{2}} e^{-\frac{1}{2\sigma^2} \sum_{y_{hatbro} \in \mathbf{y}} (y_{hatbro} - \theta_h - \alpha_a - \tau_t - \beta_b - \rho_r - \omega_o - \kappa)^2}$

9.5 Joint Posterior

Our end goal is to estimate the joint posterior distribution over all of our parameters given the evidence provided by the data: $p(\boldsymbol{\Theta}|\boldsymbol{y})$, where $\boldsymbol{\Theta}$ represents all of our parameters. Using Bayes rule, we see that the posterior may be written as a combination of the likelihood of the data and our prior probability distributions. $p(\boldsymbol{\Theta}|\boldsymbol{y}) = \frac{L(\boldsymbol{y}|\boldsymbol{\Theta})p(\boldsymbol{\Theta})}{p(\boldsymbol{y})} = \frac{L(\boldsymbol{y}|\boldsymbol{\Theta})p(\boldsymbol{\Theta})}{\int \cdots \int L(\boldsymbol{y}|\boldsymbol{\Theta})p(\boldsymbol{\Theta})d\boldsymbol{\Theta}}$. Because the normalizing constant in the denominator of this quantity involves integrating over all $\boldsymbol{\Theta}$s it can be difficult to compute. Fortunately, using Gibb's sampling to get samples from the joint posterior does not require being able to compute the normalizing constant. We can drop the constant and calculate numerator of our joint posterior, which we will call **pn**. Recall that because the denominator of the posterior is a constant, **pn** is proportional to the posterior distribution. We will similarly drop any other constants we find as we derive **pn**. Finally, because of machine precision issues when working with small probabilities, it is most useful to work directly with the logarithm of **pn**.

$$ln(\mathbf{pn}) = ln(L(\mathbf{y}|\boldsymbol{\Theta})p(\boldsymbol{\Theta}))$$

Now insert our own parameter names.

$$= ln(L(\mathbf{y}|\sigma^2, \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{\beta}, \boldsymbol{\rho}, \boldsymbol{\omega}, \kappa)p(\sigma^2, \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{\beta}, \boldsymbol{\rho}, \boldsymbol{\omega}, \kappa))$$

Because our parameters are all independent of one another, we can write their joint prior probabilities as a product of individual prior probabilities.

$$= ln(L(\mathbf{y}|\sigma^2, \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{\beta}, \boldsymbol{\rho}, \boldsymbol{\omega}, \kappa)p(\sigma^2) \prod_{h=1}^{H} p(\theta_h) \prod_{a=1}^{A} p(\alpha_a) \prod_{t=1}^{T} p(\tau_t)$$
$$\prod_{b=1}^{B} p(\beta_b) \prod_{r=1}^{R} p(\rho_r) \prod_{o=1}^{O} p(\omega_o)p(\kappa))$$

Distribute the logarithm.

$$= ln(L(\mathbf{y}|\sigma^2, \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{\beta}, \boldsymbol{\rho}, \boldsymbol{\omega}, \kappa))$$
$$+ ln(p(\sigma^2))$$
$$+ \sum_{h=1}^{H} ln(p(\theta_h))$$
$$+ \sum_{a=1}^{A} ln(p(\alpha_a))$$
$$+ \sum_{t=1}^{T} ln(p(\tau_t))$$
$$+ \sum_{b=1}^{B} ln(p(\beta_b))$$
$$+ \sum_{r=1}^{R} ln(p(\rho_r))$$

$+ \sum_{o=1}^{O} ln(p(\omega_o))$

$+ ln(p(\kappa))$

Now substitute the numerical form of the likelihood and priors.

$= ln((2\pi)^{-\frac{N}{2}}(\sigma^2)^{-\frac{N}{2}} e^{-\frac{1}{2\sigma^2}\sum_{y_{hatbro}\in\mathbf{y}}(y_{hatbro}-\theta_h-\alpha_a-\tau_t-\beta_b-\rho_r-\omega_o-\kappa)^2})$

$+ ln(\frac{1}{\Gamma(50)50^{50}}\sigma^{2(50-1)}e^{-\frac{\sigma^2}{50}})$

$+ \sum_{h=1}^{H} ln\left((2\pi * (\frac{40}{3})^2)^{-\frac{1}{2}} e^{-\frac{(\theta_h-0)^2}{2*(\frac{40}{3})^2}}\right)$

$+ \sum_{a=1}^{A} ln\left((2\pi * (\frac{40}{3})^2)^{-\frac{1}{2}} e^{-\frac{(\alpha_a-0)^2}{2*(\frac{40}{3})^2}}\right)$

$+ \sum_{t=1}^{T} ln\left((2\pi * (\frac{40}{3})^2)^{-\frac{1}{2}} e^{-\frac{(\tau_t-0)^2}{2*(\frac{40}{3})^2}}\right)$

$+ \sum_{b=1}^{B} ln\left((2\pi * (\frac{40}{3})^2)^{-\frac{1}{2}} e^{-\frac{(\beta_b-0)^2}{2*(\frac{40}{3})^2}}\right)$

$+ \sum_{r=1}^{R} ln\left((2\pi * (\frac{40}{3})^2)^{-\frac{1}{2}} e^{-\frac{(\rho_r-0)^2}{2*(\frac{40}{3})^2}}\right)$

$+ \sum_{o=1}^{O} ln\left((2\pi * (\frac{40}{3})^2)^{-\frac{1}{2}} e^{-\frac{(\omega_o-0)^2}{2*(\frac{40}{3})^2}}\right)$

$+ ln\left((2\pi * (\frac{50}{3})^2)^{-\frac{1}{2}} e^{-\frac{(\kappa-90)^2}{2*(\frac{50}{3})^2}}\right)$

Distribute logarithms deeper into terms and drop additive constants.

$\propto -\frac{N}{2}ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{y_{hatbro}\in\mathbf{y}}(y_{hatbro} - \theta_h - \alpha_a - \tau_t - \beta_b - \rho_r - \omega_o - \kappa)^2$

$+ 49ln(\sigma^2) - \frac{\sigma^2}{50}$

$- \sum_{h=1}^{H} \frac{\theta_h^2}{2(\frac{40}{3})^2}$

$- \sum_{a=1}^{A} \frac{\alpha_a^2}{2(\frac{40}{3})^2}$

$- \sum_{t=1}^{T} \frac{\tau_t^2}{2(\frac{40}{3})^2}$

$- \sum_{b=1}^{B} \frac{\beta_b^2}{2(\frac{40}{3})^2}$

$- \sum_{r=1}^{R} \frac{\rho_r^2}{2(\frac{40}{3})^2}$

$- \sum_{o=1}^{O} \frac{\omega_o^2}{2(\frac{40}{3})^2}$

$- \frac{(\kappa-90)^2}{2(\frac{50}{3})^2}$

9.6 Complete Conditionals

Now it remains to derive the complete conditional distributions of each parameter. A complete conditional distribution over parameter Θ represents the probability of that parameter given the data and the value of every other parameter in the graph, and is necessary for Gibb's sampling to function correctly. It turns out that we can use **pn**, which we have already calculated, to derive the complete conditional of each parameter simply by treating all variables except the parameter of interest as constants, and dropping as many of these constants as possible. Because **pn** is not normalized, and complete conditionals are defined as proper probability distributions, we symbolically add to each complete conditional the constant c that would correctly normalize it. However, this is only for form's sake, since Gibb's sampling does not require normalized complete conditionals. Again, because of machine precision issues, we express our conditionals in log space.

$$\left[\sigma^2\right] = (49 - \frac{N}{2})ln(\sigma^2) - \frac{\sigma^2}{50} - \frac{1}{2\sigma^2}\sum_{\mathbf{y}}(y_{hatbro} - \theta_h - \alpha_a - \tau_t - \beta_b - \rho_r - \omega_o - \kappa)^2 + c$$

$$[\alpha_a] = -\frac{\alpha_a^2}{2(\frac{40}{3})^2} - \frac{1}{2\sigma^2}\sum_{\mathbf{y}}(y_{hatbro} - \theta_h - \alpha_a - \tau_t - \beta_b - \rho_r - \omega_o - \kappa)^2 + c$$

$$[\theta_h] = -\frac{\theta_h^2}{2(\frac{40}{3})^2} - \frac{1}{2\sigma^2}\sum_{\mathbf{y}}(y_{hatbro} - \theta_h - \alpha_a - \tau_t - \beta_b - \rho_r - \omega_o - \kappa)^2 + c$$

$$[\beta_b] = -\frac{\beta_b^2}{2(\frac{40}{3})^2} - \frac{1}{2\sigma^2}\sum_{\mathbf{y}}(y_{hatbro} - \theta_h - \alpha_a - \tau_t - \beta_b - \rho_r - \omega_o - \kappa)^2 + c$$

$$[\tau_t] = -\frac{\tau_t^2}{2(\frac{40}{3})^2} - \frac{1}{2\sigma^2}\sum_{\mathbf{y}}(y_{hatbro} - \theta_h - \alpha_a - \tau_t - \beta_b - \rho_r - \omega_o - \kappa)^2 + c$$

$$[\rho_r] = -\frac{\rho_r^2}{2(\frac{40}{3})^2} - \frac{1}{2\sigma^2}\sum_{\mathbf{y}}(y_{hatbro} - \theta_h - \alpha_a - \tau_t - \beta_b - \rho_r - \omega_o - \kappa)^2 + c$$

$$[\omega_o] = -\frac{\omega_o^2}{2(\frac{40}{3})^2} - \frac{1}{2\sigma^2}\sum_{\mathbf{y}}(y_{hatbro} - \theta_h - \alpha_a - \tau_t - \beta_b - \rho_r - \omega_o - \kappa)^2 + c$$

$$[\kappa] = -\frac{(\kappa - 90)^2}{2(\frac{50}{3})^2} - \frac{1}{2\sigma^2}\sum_{\mathbf{y}}(y_{hatbro} - \theta_h - \alpha_a - \tau_t - \beta_b - \rho_r - \omega_o - \kappa)^2 + c$$

References

[1] J. Baldridge and M. Osborne. Active Learning and the Total Cost of Annotation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 9–16, 2004.

[2] S. Bangalore and A.K. Joshi. Supertagging: An Approach to Almost Parsing. *Computational Linguistics*, 25(2):237–265, 1999.

[3] B. Beigman Klebanov and E. Beigman. From Annotator Agreement to Noise Models. *Computational Linguistics*, 35(4):495–503, 2009.

[4] J.O. Berger and D.A. Berry. Statistical Analysis and the Illusion of Objectivity. *American Scientist*, 76(2):159–165, 1988.

[5] T. Brants and O. Plaehn. Interactive Corpus Annotation. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, 2000.

[6] M. Carmen, P. Felt, R. Haertel, D. Lonsdale, P. McClanahan, O. Merkling, E. Ringger, and K. Seppi. Tag Dictionaries Accelerate Manual Annotation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, 2010.

[7] M.W. Chang, L. Ratinov, and D. Roth. Guiding Semi-supervision with Constraint-driven Learning. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 280–287, 2007.

[8] M.W. Chang, L. Ratinov, N. Rizzolo, and D. Roth. Learning and Inference with Constraints. In *Proceedings of the 23nd Conference on Artificial Intelligence*, pages 1513–1518. AAAI, 2008.

[9] F.D. Chiou, D. Chiang, and M. Palmer. Facilitating Treebank Annotation Using a Statistical Parser. In *Proceedings of the First International Conference on Human Language Technology Research*, pages 1–4, 2001.

[10] J. Cowie and W. Lehnert. Information Extraction. *Communications of the ACM*, 39(1):80–91, 1996.

[11] M.K. Cowles and B.P. Carlin. Markov Chain Monte Carlo Convergence Diagnostics: a Comparative Review. *Journal of the American Statistical Association*, 91(434):883–904, 1996.

[12] A. Culotta and A. McCallum. Reducing Labeling Effort for Structured Prediction Tasks. In *Proceedings of the 20th Conference on Artificial Intelligence*, pages 746–751. AAAI, 2005.

[13] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 168–175, 2002.

[14] G. Druck, G. Mann, and A. McCallum. Learning from Labeled Features using Generalized Expectation Criteria. In *Proceedings of the 31st Annual International Conference on Research and Development in Information Retrieval*, pages 595–602. ACM, 2008.

[15] P. Felt, O. Merkling, M. Carmen, E. Ringger, W. Lemmon, K. Seppi, and R. Haertel. CCASH: A Web Application Framework for Efficient Distributed Language Resource Development. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, 2010.

[16] P. Felt, E. Ringger, K. Seppi, K. Heal, R. Haertel, and D. Lonsdale. First Results in a Study Evaluating Pre-labeling and Correction Propagation for Machine-Assisted Syriac Morphological Analysis. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, 2012.

[17] D. Ferrucci. Building Watson: An Overview of DeepQA for the Jeopardy! Challenge. In *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques*, pages 1–2, 2010.

[18] K. Ganchev, F. Pereira, M. Mandel, S. Carroll, and P. White. Semi-automated Named Entity Annotation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics: Linguistic Annotation Workshop*, pages 53–56, 2007.

[19] K. Ganchev, J.V. Graça, J. Blitzer, and B. Taskar. Multi-view Learning over Structured and Non-identical Outputs. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 204–211. AAAI, 2008.

[20] A. Gelman. *Bayesian Data Analysis*. CRC press, 2004.

[21] R. Haertel, E. Ringger, K. Seppi, J. Carroll, and P. McClanahan. Assessing the Costs of Sampling Methods in Active Learning for Annotation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 65–68, 2008.

[22] R. Haertel, K.D. Seppi, E.K. Ringger, and J.L. Carroll. Return on Investment for Active Learning. In *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems: Workshop on Cost-sensitive Learning*, 2008.

[23] K. Hale, M. Krauss, L.J. Watahomigie, A.Y. Yamamoto, C. Craig, L.V.M. Jeanne, and N.C. England. Endangered Languages. *Language*, 68(1):1–42, 1992.

[24] K. Heal. Personal communication, 2011.

[25] G.A. Kiraz. Automatic Concordance Generation of Syriac Texts. *Orientalia Christiana Analecta*, (247):461–475, 1994.

[26] A. Kittur, E.H. Chi, and B. Suh. Crowdsourcing User Studies with Mechanical Turk. In *Proceedings of the 26th Annual Conference on Human Factors in Computing Systems*, pages 453–456. ACM, 2008.

[27] T. Kristjansson, A. Culotta, P. Viola, and A. McCallum. Interactive Information Extraction with Constrained Conditional Random Fields. In *Proceedings of the 19th Conference on Artificial Intelligence*, pages 412–418. AAAI, 2004.

[28] C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[29] M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

[30] P. McClanahan, G. Busby, R. Haertel, K. Heal, D. Lonsdale, K. Seppi, and E. Ringger. A Probabilistic Morphological Analyzer for Syriac. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 810–820. ACL, 2010.

[31] J. Menke and T.R. Martinez. Using Permutations Instead of Student's t Distribution for P-values in Paired-difference Algorithm Comparisons. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1331–1335. IEEE, 2004.

[32] T. Morton and J. LaCivita. WordFreak: an Open Tool for Linguistic Annotation. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Demonstrations*, pages 17–18, 2003.

[33] C. Müller and M. Strube. Multi-level Annotation of Linguistic Data with MMAX2. *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, 3:197–214, 2006.

[34] T. Naseem, H. Chen, R. Barzilay, and M. Johnson. Using Universal Linguistic Knowledge to Guide Grammar Induction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244. ACL, 2010.

[35] G. Ngai and D. Yarowsky. Rule Writing or Annotation: Cost-efficient Resource Usage for Base Noun Phrase Chunking. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 117–125, 2000.

[36] P.V. Ogren. Knowtator: A Protégé Plug-in for Annotated Corpus Construction. In *Proceedings of the 8th Annual Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Demonstrations*, pages 273–275, 2006.

[37] B. Pang and L. Lee. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[38] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs Up?: Sentiment Classification using Machine Learning Techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 79–86. ACL, 2002.

[39] E. Ringger, P. McClanahan, R. Haertel, G. Busby, M. Carmen, J. Carroll, K. Seppi, and D. Lonsdale. Active Learning for Part-of-speech Tagging: Accelerating Corpus Annotation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics: Linguistic Annotation Workshop*, pages 101–108, 2007.

[40] E. Ringger, M. Carmen, R. Haertel, K. Seppi, D. Lonsdale, P. McClanahan, J. Carroll, and N. Ellison. Assessing the Costs of Machine-assisted Corpus Annotation Through a User Study. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, 2008.

[41] B. Settles. Active Learning Literature Survey. *Computer Sciences Technical Report 1648. University of Wisconsin, Madison*, 2010.

[42] B. Settles, M. Craven, and L. Friedland. Active Learning with Real Annotation Costs. In *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems: Workshop on Cost-Sensitive Learning*, pages 1069–1078, 2008.

[43] W.M. Soon, H.T. Ng, and D.C.Y. Lim. A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics*, 27(4):521–544, 2001.

[44] K. Tomanek and F. Olsson. A Web Survey on the Use of Active Learning to Support Annotation of Text Data. In *Proceedings of the 10th Annual Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Workshop on Active Learning for Natural Language Processing*, pages 45–48, 2009.

[45] K. Tomanek, J. Wermter, and U. Hahn. Efficient Annotation with the Jena ANnotation Environment (JANE). In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics: Linguistic Annotation Workshop*, pages 9–16, 2007.

[46] E. Tov and N.B. Reynolds. *The Dead Sea Scrolls Electronic Library*. Brigham Young University and the Neal A. Maxwell Institute for Religious Scholarship, 2006.

[47] L. Von Ahn and L. Dabbish. Designing Games With a Purpose. *Communications of the ACM*, 51(8):58–67, 2008.

[48] W. Wright. *Apocryphal Acts of the Apostles*. Williams and Norgate, 1871.