2015-11-01

Interactive Machine Assistance: A Case Study in Linking Corpora and Dictionaries

Kevin P. Black
Brigham Young University - Provo

Interactive Machine Assistance:

A Case Study in Linking Corpora and Dictionaries

Kevin P. Black

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

Master of Science

Eric K. Ringger, Chair
Kevin Seppi
Quinn Snell

Department of Computer Science

Brigham Young University

November 2015

ABSTRACT

Interactive Machine Assistance:
A Case Study in Linking Corpora and Dictionaries

Kevin P. Black
Department of Computer Science, BYU
Master of Science

Machine learning can provide assistance to humans in making decisions, including linguistic decisions such as determining the part of speech of a word. Supervised machine learning methods derive patterns indicative of possible labels (decisions) from annotated example data. For many problems, including most language analysis problems, acquiring annotated data requires human annotators who are trained to understand the problem and to disambiguate among multiple possible labels. Hence, the availability of experts can limit the scope and quantity of annotated data. Machine-learned pre-annotation assistance, which suggests probable labels for unannotated items, can enable expert annotators to work more quickly and thus to produce broader and larger annotated resources more cost-efficiently. Yet, because annotated data is required to build the pre-annotation model, bootstrapping is an obstacle to utilizing pre-annotation assistance, especially for low-resource problems where little or no annotated data exists. Interactive pre-annotation assistance can mitigate bootstrapping costs, even for low-resource problems, by continually refining the pre-annotation model with new annotated examples as the annotators work. In practice, continually refining models has seldom been done except for the simplest of models which can be trained quickly.

As a case study in developing sophisticated, interactive, machine-assisted annotation, this work employs the task of corpus-dictionary linkage (CDL), which is to link each word token in a corpus to its correct dictionary entry. CDL resources, such as machine-readable dictionaries and concordances, are essential aids in many tasks including language learning and corpus studies. We employ a pipeline model to provide CDL pre-annotations, with one model per CDL sub-task. We evaluate different models for lemmatization, the most significant CDL sub-task since many dictionary entry headwords are usually lemmas. The best performing lemmatization model is a hybrid which uses a maximum entropy Markov model (MEMM) to handle unknown (novel) word tokens and other component models to handle known word tokens. We extend the hybrid model design to the other CDL sub-tasks in the pipeline. We develop an incremental training algorithm for the MEMM which avoids wasting previous computation as would be done by simply retraining from scratch. The incremental training algorithm facilitates the addition of new dictionary entries over time (i.e., new labels) and also facilitates learning from partially annotated sentences which allows annotators to annotate words in any order. We validate that the hybrid model attains high accuracy and can be trained sufficiently quickly to provide interactive pre-annotation assistance by simulating CDL annotation on Quranic Arabic and classical Syriac data.

Keywords: interactive machine assistance, machine-assisted annotation, corpus-dictionary linkage annotation, supervised machine learning, string transduction, hybrid probabilistic models, low-resource languages, low-resource language settings, Arabic, Quran, Syriac, New Testament

ACKNOWLEDGMENTS

As with any major project, the development of this thesis has been a community effort. For the work to come to fruition it has required spiritual inspiration, emotional support, social mentoring, and technical collaboration.

First, I required spiritual inspiration and encouragement from Heavenly Father to do this work. I started working on my masters degree with the assurance that pursuing it was a good thing to do. In the course of my studies and in doing the thesis, Heavenly Father has answered many personal prayers requesting understanding, motivation, and time to be able to do the work with integrity and enthusiasm. Almost every lab meeting began with prayer to thank God that we could work and study at Brigham Young University and to request inspiration and strength to be able to bless God's children through our work and studies. I know that those prayers have been answered and will continue to be answered.

Second, I was able to complete this thesis because of the emotional support of my family and friends. My wife, Amanda, has always expressed unfailing faith that the work would all come together. She has been exceedingly patient again and again as the thesis focus and "deadlines" shifted, and she has taken care of chores and meals so that I would have more time to code and to write. The recent birth of my daughter, Elisabeth, provided renewed drive to finish the thesis. I appreciate the encouragement from my family, in-laws, and friends. Their interest in the project and patience with me as I learned to explain it in simple terms helped me to remain energized and excited about the research.

Third, the vision, scope, and polish of the thesis is due to patient and persistent social mentoring. Dr. Eric Ringger has provided the vision for researching machine assisted annotation, and the leadership necessary to bring computer science, linguistic, and Syriac scholarship together to yield insight to human and machine collaboration. He was kind to enlist me in the ongoing project and to let me take control of the project for a season. Dr. Ringger has been unfailingly patient as the scope of the thesis had to be scaled back and the coding and experimentation took longer than expected. Dr. Kevin Seppi has always sought to keep the scope of the thesis small enough to be manageable and to provide interesting and compelling results. He has pushed me to explain things simply and clearly and to be assertive and precise both in speaking and in writing. Dr. Deryle Lonsdale has provided linguistic insight and explanations for language in general and Arabic in particular. The research group has benefited from his viewpoint and experience as we have sought

to present the research within the language research community. Dr. Kristian Heal has graciously taught me the basics of Syriac pronunciation and grammar and has enthusiastically answered my questions regarding Syriac studies. He has been very patient working with a bunch of computer scientists, and yet again will have to wait for the development of a functioning machine assisted annotation platform to aid in the Syriac Electronic Corpus project. Paul Felt has been an optimistic and understanding peer mentor to me as I have learned to conduct and present research. He was always willing to talk about and help me sort out ideas that I encountered in my research.

I appreciate the further social mentoring that came from fellow lab members and from co-workers at my various internships and full-time employment. Thank you to all of the lab members who have been engaged in the meetings, both to share research and to to engage with me in my research by asking questions, offering feedback, and being a good audience with which to practice my research presentations. Thank you to Paul Felt, Nozomu Okuda, Nick Wilson, Joshua Reynolds, Ruger Dutton, Jeff Lund, Hito Matsushita, Kevin Cook, Chris Tensmeyer, Craig Jacobson, Schuyler Goodman, Daniel Ricks, Ethan Garofolo, Joey Cozza, Jared Foresyth, Robbie Haertel, Dan Walker, Bill Lund, and Andrew McNabb. Thank you to all of my co-workers who expressed interest in the project and encouraged me to pursue my masters degree. Several co-workers have been particularly influential. Pablo Riboldi was a strong advocate for further education and was understanding when I needed to leave work to be a full-time student. Craig McClanahan was enthusiastic about the research and connected me with his son Peter who was one of the first students to work on the machine assistance for Syriac project. Pat Schone was a great mentor who engaged me in his various research projects, trusted my judgment and abilities, and demonstrated the importance of getting functional prototype systems built sooner than later. Richard Andrew has been a great carpool driver, letting me talk about my thesis and work on editing while we are driving to and from work.

Fourth, the code supporting the thesis is a product of technical collaboration. I was able to perform my research because of the code written by Peter McClanahan, Robbie Haertel, George Busby, Paul Felt, and others. Though our personal coding styles are different, I was glad that I did not have to implement maxent Markov models and data processing tools from scratch. I hope that my additions to the code base will continue to contribute to future research. Software development requires IT support, for which I thank Paul Felt, the resident Linux guru and system admin for the natural language processing lab. Also, the crowdsourcing study summarized in the introduction to the thesis was conducted by Paul Felt. I appreciate the technical and pedagogical expertise of the professors I worked with in my masters study classes: Dr. Eric

Ringger, Dr. Kevin Seppi, Dr. Tony Martinez, Dr. Phil Windley, Dr. Dan Olsen, and Dr. Mike Goodrich.

I thank the professors that encouraged me to pursue a masters degree. Dr. Eric Ringger, Dr. Tony Martinez, and Dr. David Embley were persuasive in advocating graduate school in general and graduate school at BYU in particular. Dr. Eric Ringger, Dr. Daniel Zappala, and Prof. Craig Laurence wrote the letters of recommendation for my graduate school application; though I have not seen the letters I appreciate their faith and confidence in my ability to do effective research.

I thank the computer science department assistants and secretaries that ensured that my studies were successful. Jen Bonnet has helped me know and successfully navigate the graduate studies logistics. Jenny Thornton ensured that I was hired and paid as a research and teaching assistant. Kimberly Jenkins has always been cheerful and happy to work with me to schedule times and spaces for lab meetings.

Finally, to all those that have helped me learn leadership, assisted me to gain technical mastery of computer science, and have ultimately made this thesis a success, I say thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis addresses the problem domain of accelerating expert human annotation, specifically low-resource language annotation tasks, through the use of machine assisted annotation. Annotated data informs decision making, and should therefore be as correct and as consistent as possible so analyses will be valid and established policies will be effective. As examples: (1) purchase histories assist in determining what products to recommend to individual customers, and (2) text corpora with authorship dates per document and part of speech tags for the words facilitate studying the use and evolution of words over time. Acquiring quality annotated data often requires hiring experts to do the annotation, especially for low-resource language annotation tasks for which little or no annotated data already exists. Annotation tasks which require expert annotation are limited in their quantity and scope because remuneration and opportunity costs limit expert involvement. Accelerating the pace at which expert annotators are able to work allows expert-annotated data to be acquired more cost-efficiently. Machine-learned pre-annotation assistance, which suggests probable labels for unannotated data items, can accelerate the development of expert-annotated data sets (cf. Ringger et al., 2008; Carmen et al., 2010; Felt et al., 2013). However, machine-learned pre-annotation assistance requires annotated data in the first place from which to learn a sufficiently accurate predictive model. Our research goal is to develop pre-annotation machine assistance methods which do not require an initial annotated data set and which improve as the annotators are working on the corpus so that the methods can therefore be employed for even low-resource language annotation tasks.

Before introducing our solution to accelerate expert annotation, including in low-resource settings, we summarize methods for acquiring annotated data to further demonstrate what we mean by "expert annotation". Depending on the difficulty of the annotation task, annotated data can be acquired via automation, crowdsourcing, hiring expert annotators, or some combination of those approaches (e.g., Basile et al., 2012). Automation works well for monitoring user actions, such as purchases, or when sufficient annotated data

exists with which to train a high quality machine-learned model and then employ the model to supply annotations for new data. Crowdsourcing is the solicitation of redundant opinions over the Internet from many people with an average set of skills (cf. Quinn and Bederson, 2011; Yuen et al., 2011). Crowdsourcing is effective for simple tasks that require human judgement or that are easy for humans to do but complex for computer algorithms, such as object identification in images. Crowdsourcing may be conducted via micro-job markets like Amazon Mechanical Turk or CrowdFlower, or may be engineered into social games (e.g., von Ahn, 2006; Venhuizen et al., 2013), or may be built into web sites to verify human users, such as with the reCAPTCHA image labeling system (von Ahn et al., 2008). Hiring experts to annotate is required for specialized annotation tasks for which the experts are trained to understand the data and/or to disambiguate among multiple possible annotations. For example, crowdsourcing is insufficient for identifying the lemma of Syriac words, inasmuch as none of the 3,051 CrowdFlower workers that attempted our Syriac language lemmatization task in the fall of 2013 performed at the required minimum accuracy of 75% on the trial job (first reported by Black et al., 2014). Many language analysis problems require hiring both domain specialists and linguists since they have complimentary expertise (cf. Tsuboi et al., 2008).

The remainder of this chapter introduces the key ideas relevant to our approach to accelerating expert annotation including in low-resource settings. First, in Section 1.1, we discuss the desired solution to accelerate expert annotation, namely interactive pre-annotation assistance. Second, in Section 1.2, we introduce the corpus-dictionary linkage task for which we will develop interactive pre-annotation assistance mechanisms. We then summarize sequence tagging in Section 1.3 which is the problem formulation we use to provide context-sensitive pre-annotations and then finish in Section 1.4 with an overview of the thesis.

1.1 Desired Solution: Interactive Pre-annotation Assistance

Traditional pre-annotation assistance can accelerate expert annotation but requires annotated data to bootstrap the assistance. Pre-annotation assistance accelerates expert annotation when, for a given data item (e.g., word token), the predictive model ranks the correct label high in its list of suggestions, thus allowing an annotator to validate and accept the correct label more quickly than entering it manually (cf. Ringger et al., 2008; Carmen et al., 2010; Felt et al., 2013). Traditionally, pre-annotation assistance is employed in a batch workflow where a portion of data (e.g., document) is automatically annotated and then manually reviewed and corrected; then (optionally) the pre-annotation model is retrained with the new annotations, and

the process repeats. For text annotation, the batch workflow is supported by a variety of collaborative, web-based annotation frameworks such as Cost Conscious Annotation Supervised by Humans (CCASH) (Felt et al., 2010), General Architecture for Text Engineering (GATE) (Cunningham et al., 2011)—specifically GATE Teamware (Bontcheva et al., 2013)—brat (Stenetorp et al., 2012), and WebAnno (Yimam et al., 2014). For pre-annotation assistance to be effective the model must provide high-quality annotation suggestions which in turn requires a sufficiently diverse and/or large annotated data set with which to train the model. Hence, for low-resource problems where little or no annotated data exists, the limited availability of experts to produce an annotated data set is still a significant obstacle to beneficially employ pre-annotation assistance.

In theory, *interactive* pre-annotation assistance, which updates the model as the annotators work, eliminates the need to annotate a data set to train (bootstrap) the initial model. Interactive pre-annotation assistance shortens the feedback loop in the annotation workflow in two ways: first, by immediately updating pre-annotations in light of new manual annotations and corrections, and second, by regularly and quickly retraining the model with the expanded and refined set of annotated data (Culotta et al., 2006). To be interactive, model training and prediction times need to be sufficiently short so that it is apparent to the annotators that the system is learning from their annotations and so that the annotators need not wait for the system to update before they can proceed with their work. Because of the tight, interactive feedback loop between the annotators and the pre-annotation model, the model improves and provides better assistance during the course of the annotation project and the annotators work more quickly and consistently. Consequently, interactive pre-annotation assistance can enable more rapid development of expert-annotated data sets for low-resource problems because annotators can leverage pre-annotation assistance beginning with whatever annotated resources they do have, even starting with no such resources.

1.2 Task of Interest: Corpus-Dictionary Linkage

We employ the annotation task of corpus-dictionary linkage (CDL) as a case study in developing practical interactive pre-annotation assistance. The objective of the CDL task is to annotate each word token in a text corpus with a link to a dictionary entry that defines or describes the word token. When there is no suitable dictionary entry for a given word token a new entry must be created. Linking a corpus to a dictionary thus simultaneously indexes the corpus by gathering related tokens under appropriate headwords

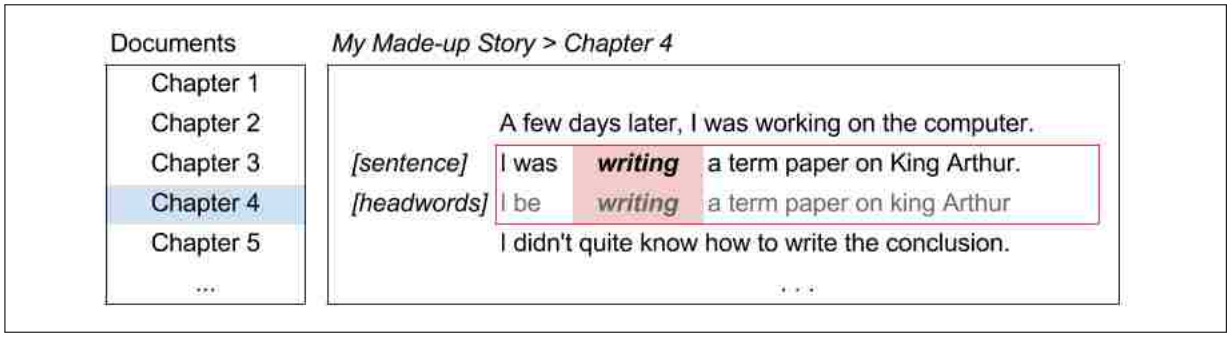
and expands the dictionary with new entries and usage examples. Some lexicographers perform CDL to produce dictionaries (e.g., Kilgarriff, 2005) or add usage examples to existing dictionaries (e.g., Caselli et al., 2014). Similarly, concordance editors could use CDL data to compile a concordance for a selected corpus. Corpus-dictionary linked resources include concordances, dictionaries with usage examples, and corpora annotated with lemmas, word senses, or glosses. CDL resources are essential aids in many tasks including language learning and corpus studies.

In order to ground the remaining discussion of machine assistance for linking corpora and dictionaries, we sketch a tool for CDL annotation. Figure 1.1 sketches important user interface components. Annotators may navigate through the corpus by browsing (Figure 1.1a) or by searching for a specific word type (Figure 1.1b); data is shown in context with suggested dictionary headwords for each word token in the sentence or sentences in focus. An annotator may select and annotate any word token or group of word tokens of the same word type The facility to select and annotate multiple word tokens in the keyword-in-context (KWIC) search view, even without machine assistance, naturally accelerates manual annotation (e.g., Tsuboi et al., 2008). When a word token is selected (shaded in Figures 1.1a and 1.1b), a more comprehensive list of pre-annotation suggestions is shown (Figure 1.1c) and broken down into different categories depending on how the suggested dictionary entries relate to the word token. The annotator may accept a suggested dictionary entry using the check mark or plus symbol buttons, which may require creating a new dictionary entry for novel headwords (plus symbol button), or may create a dictionary entry not suggested by the system ("New" button), or may browse the dictionary (Figure 1.1d) and apply an entry to the currently selected word token(s) ("apply" buttons).

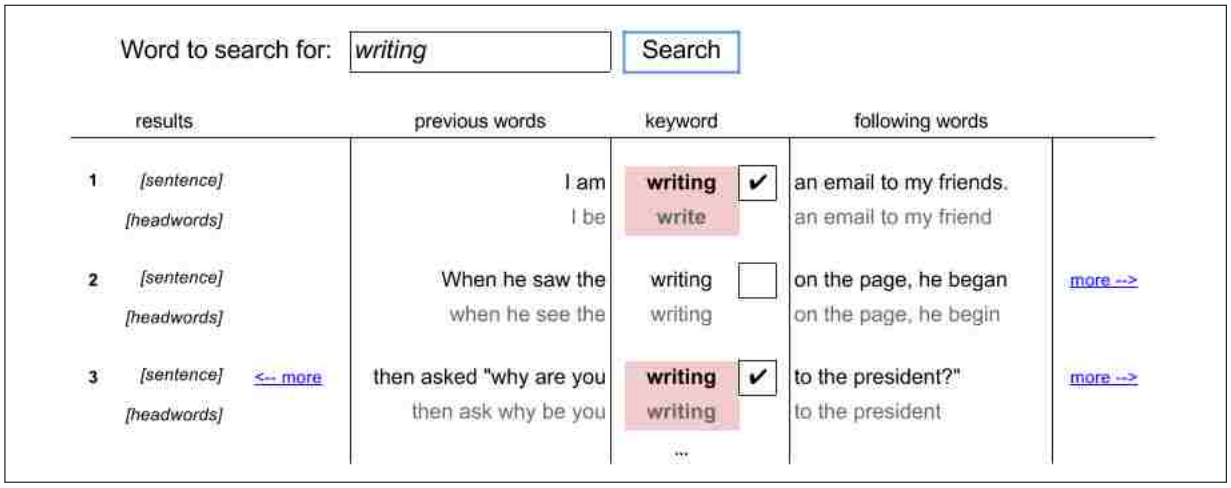
1.3 Sequence Tagging for Context-sensitive Pre-annotation

We formulate the CDL annotation task as a sequence tagging problem so that pre-annotation assistance may be context-sensitive. Formulating CDL as a sequence tagging problem, the goal is to predict a dictionary link (tag) for each word token in a given sentence, thus forming a tag sequence parallel to the token sequence. Sequence tagging models are sensitive to both static and dynamic context: that is, the ranking of candidate dictionary links for each word token is influenced by features (e.g., character patterns) from the word token itself (static), and by features from surrounding word tokens (static), and by predicted dictionary links for surrounding word tokens (dynamic), and by combinations of such static and dynamic features.

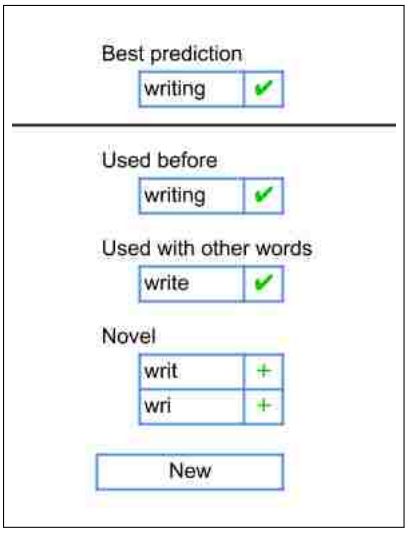
(a) Browse text. Selected verse highlighted. Selected token shaded.



(b) Search text. Results shown keyword-in-context (KWIC). Selected tokens shaded and checkboxes checked.



(c) Suggested dictionary links for word token(s) which are selected in the text.



(d) Dictionary entry. Facilitates browsing the dictionary and showing detail of a suggested dictionary link.

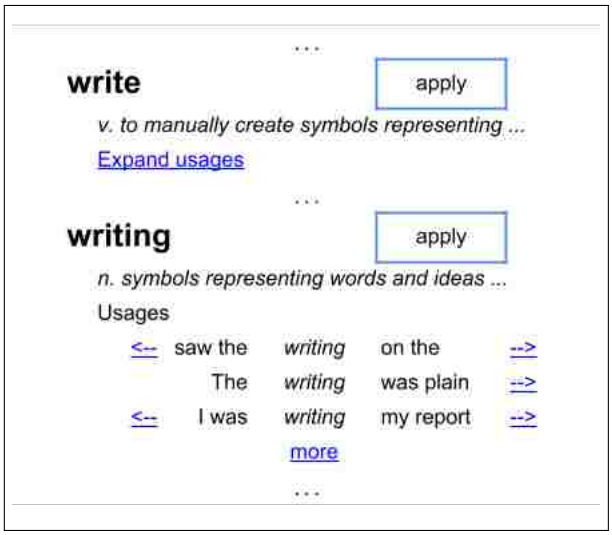


Figure 1.1: User interface component sketches.

The advantage of context-sensitive ranking is that a word token's context may help to disambiguate among multiple candidate dictionary links. The simplest model for interactive pre-annotation assistance memorizes the frequency with which labels are associated with each data item type in the training data and then uses that frequency index to rank annotation suggestions. The memorizer is trivial to retrain but is not context-sensitive like sequence tagging models, and so lacks the ability to disambiguate among multiple candidate dictionary links. The sequence tagging models we employ are data-driven, using only features derived from tokens and dictionary links rather than features from expert-engineered knowledge sources, and can thus be applied to any language including low-resource languages.

1.4 Thesis Overview

The key contribution of the thesis is the development of interactive pre-annotation models for the CDL sequence tagging task. Interactive pre-annotation assistance has been implemented for classification tasks (e.g., Settles, 2011; Clawson and Barrett, 2015) which predict a single tag for the input, but not for sequence tagging or other structured prediction tasks that have dependencies between the tags. This is likely because the time to train sequence taggers is not trivial because of the (potentially) large number of features for which a weight must be learned. Additionally, the number of labels (dictionary entries) and the number of features grows throughout the course of annotation. An intuitive way to reduce retraining time and handle new labels and features is to retrain the model incrementally: start with the previous instance of the model and adjust its parameters given new annotated training examples. A further challenge is to learn from partially annotated sentences, which is a consequence of affording annotators the flexibility to annotate tokens in any order they want, including, and especially, by selecting multiple tokens of the same word type in a keyword-in-context (KWIC) control and then annotating all of the selected tokens at once (see Figure 1.1b). Thus, more concretely, we develop a solution to train sequence tagging models designed for corpus-dictionary linkage annotation in an incremental fashion and with partially annotated data. The goal is to achieve good prediction quality and short training times so that the models can be employed for interactive assistance and could thus accelerate expert annotation. We ultimately validate that our solution is suitably interactive, even for low-resource settings, by simulating CDL annotation for the Semitic languages of classical Syriac and Quranic Arabic.

We proceed to present the thesis by reviewing the corpus-dictionary linkage task in greater detail

in Chapter 2. We review other work related to interactive pre-annotation assistance and sequence tagging in Chapter 3. The thesis statement is given in Chapter 4. The Arabic and Syriac data sets we use in our experiments are discussed in Chapter 5. We present our evaluation of pre-annotation models for the stem-to-lemma linkage CDL sub-task in Chapter 6. We develop an incremental training algorithm for a context-sensitive CDL model and validate that it is suitable to support interactive pre-annotation assistance in Chapter 7. We summarize conclusions and suggest promising future work in 8.

Chapter 2

Corpus-Dictionary Linkage

In this chapter we discuss the subtask components of corpus-dictionary linkage (CDL) and existing work related to those subtasks for classical Syriac and Quranic Arabic. We use the CDL task as a case study in accelerating expert annotation and we will validate our interactive pre-annotation assistance methods by simulating CDL annotation of classical Syriac and Quranic Arabic texts. We first define the CDL task in Section 2.1 and then break it down into subtasks in Section 2.2. We devote much of our attention to the first subtask of lemmatization, since this is the most critical component to CDL, and we summarize methods and related work in Section 2.3. We briefly address the second subtask of disambiguating among dictionary entries with the same headword in Section 2.4 and then review dictionary curation in Section 2.5 which is related to the task of corpus-dictionary linkage. In Section 2.6 we summarize key decisions we make regarding our approach to implement interactive pre-annotation assistance for corpus-dictionary linkage.

2.1 CDL Task Definition

As explained in the introductory chapter, the objective of the CDL task is to annotate each word token in a text corpus with a link to a dictionary entry that defines or describes the word token. When there is no suitable dictionary entry for a given word token a new entry must be created. The CDL task is similar to the task of Wikification where instances of important keywords and entities in a document are linked to Wikipedia articles (cf. Mihalcea and Csomai, 2007), except that in CDL *every* word token is linked to a dictionary entry. Corpus-dictionary linked resources include concordances, dictionaries with usage examples, and corpora annotated with lemmas,[1] word senses, or glosses.[2] CDL resources are essential aids in many tasks including language learning and corpus studies.

[1] Lemmas are base word forms; for example "*write*" is the lemma for "*written*", "*writing*", and so on.
[2] Glosses are equivalent words in another language.

2.2 Subtasks

The CDL task can be naturally broken down into smaller sub-tasks depending on the language of interest and the structure of the dictionary. Generally speaking, the CDL task can be accomplished in two steps[3]: for a given word token, (1) identify the best dictionary headword for the word token; (2) select the best entry from the set of entries subordinate to the selected best headword. Figure 2.1 illustrates an example of CDL with these two general steps for a few word tokens from the Syriac Peshitta New Testament (PNT). Since many dictionaries employ the language's lemmas as headwords, the first step of CDL annotation consists of lemmatization. The second step entails disambiguating among multiple entries under the same headword, potentially distinguished by grammatical category (part of speech) or word sense.[4] We refer to the second step as the task of homographic headword disambiguation.

2.3 Lemmatization

The first step in CDL can be accomplished at a coarse level of granularity via lemmatization since a token's lemma, once identified, will match only a small set of homographic dictionary headwords. Lemmatization can be accomplished either directly or in two steps. **Token-to-lemma linkage**, or selecting a lemma for a given word token *directly*, has been implemented using machine-learned predictive models for languages such as Spanish, German, and Japanese (Chrupała, 2006), and for classical Syriac (Lindgren, 2011). The *two-step* approach to lemmatization, as shown in the corpus (top) portion of Figure 2.1, first decomposes each token into morphological segments, namely prefixes, stems, and suffixes, and then subsequently predicts lemmas for each stem.[5] The two-step lemmatization approach has been applied to the Arabic Quran (Dukes and Habash, 2010) using the Buckwalter finite-state morphology (Buckwalter, 2002), and to the Syriac New Testament using the SyroMorph morphological disambiguation tool (McClanahan, 2010). **Morphological segmentation** (step one) can be solved with a high level of accuracy; for example, SyroMorph's morphological segmentation method (cf. Haertel et al., 2010b) achieves about 99% accuracy overall on the Syriac New Testament. **Stem-to-lemma linkage** (step two) is more difficult; for example, SyroMorph

[3]Text without whitespace separating word tokens (e.g., Japanese) requires an additional preliminary step to segment words.

[4]Most dictionaries of Semitic languages produced in the nineteenth and twentieth centuries were organized by root, with derived forms appearing after the root entry (e.g., Brockelmann, 1895; Payne Smith, 1903). Thus, an additional step of identifying the root corresponding to a chosen entry could be considered as part of an enhanced CDL task.

[5]We define the stem to be a substring of the word token and hence it may or may not be a word itself; for example, "*writ*" is the stem for "*writing*". The lemma is always a word; e.g., "*write*".

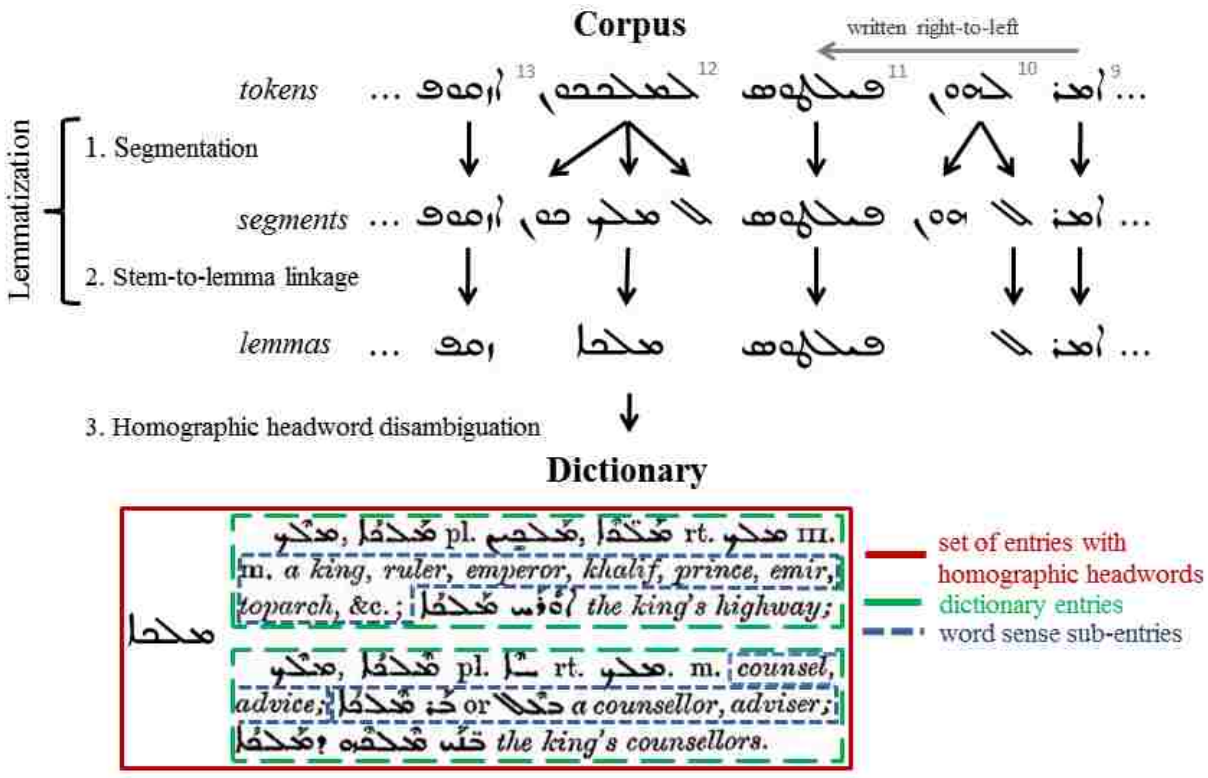


Figure 2.1: An illustration of corpus-dictionary linkage applied to text from John 19:15 of the Syriac PNT. *Corpus*: depicts two-step lemmatization: 1. Each word token is segmented into prefix, stem, and suffix. 2. Each stem is linked to a lemma. *Dictionary*: illustrates CDL variants: A lemma identifies a set of headwords (solid green line). More nuanced CDL disambiguates among homographic headwords (long-dashed red line). Still more nuanced CDL disambiguates among the multiple word senses (short-dashed blue line). Dictionary images from the Payne Smith (1903) dictionary, page 277.

achieves only 96% accuracy overall. The stem-to-lemma linkage task can be solved with string transducers, which transform one string (the input stem) into another (the output lemma). As part of this work (see Chapter 6) we have applied and evaluated several existing string transduction methods to the stem-to-lemma linkage problem, including Morfette (Chrupała, 2006; Chrupała et al., 2008), a variant of Morfette called hybrid-Morfette (McClanahan et al., 2010), and DirecTL+ (Jiampojamarn et al., 2008).

Morphological analyzers can be used to support lemmatization since the enumerated morphological analyses identify possible lemmas. Morphological analysis includes the task of "labeled morphological segmentation" (Cotterell et al., 2015) in which word tokens are segmented into morphemes and each morpheme is labeled with morphological tags, and, as appropriate for the language, morphological analysis furthermore identifies the lemma and root (i.e., base word forms) for each stem. Morphological analysers produce an unranked enumeration of candidate analyses for each word type and/or sentence. For Syriac and Arabic,

significant work has been done to build automatic morphological analyzers. Buckwalter's morphological analyzer for Arabic is perhaps the best known analyzer for a Semitic language (Buckwalter, 2002). Kiraz (2000) built a finite state morphological analyzer for Syriac and Arabic, but unfortunately the implementation is no longer available. However, since morphological analyzers often encode rules created by language experts, they are thus expensive to create and may not be available for many low-resource language settings so we do not employ them in this work.

Morphological disambiguators can also be used to support lemmatization, and in fact facilitate pre-annotation assistance. Morphological disambiguation extends morphological analysis by ranking candidate analyses based on some measure of the correctness of each analysis given the word token's context. A morphological disambiguation system can thus be used to provide annotators with ranked lemmatization suggestions. A well-known morphological disambiguator for Arabic is MADA developed by Habash and Rambow (2005). MADA builds a machine-learned classifier per morphological attribute. MADA then scores the morphological analyses produced by the ALMORGEANA morphological analyzer (Habash, 2007), based on the Buckwalter morphological analyzer, by combining the predictions and confidences of the individual morphological attribute classifiers. MADA achieves between 93% and 96% accuracy depending on the combination method and the test set, which is 4 to 6 percentage points better than the baseline approach of predicting the most frequent morphological analysis in training data. McClanahan et al. (2010) built a probabilistic morphological disambiguation system for classical Syriac called SyroMorph and report a morphological tag accuracy of 97%. SyroMorph distinguishes itself by being fully data-driven, avoiding dependence on knowledge-engineered resources such as expert-crafted morphological analyzers. This fact makes the use of SyroMorph appealing for low-resource settings. Recently, Lindgren (2011) built a morphological analyzer for classical Syriac called dkrMorph, which provides some basic disambiguation functionality by ranking candidate analyses according to frequency of occurrence in a training set. Despite the simplicity of its ranking algorithm, dkrMorph reports 94.42% accuracy on a held-out set of token-to-lemma links. Breaking down that result, dkrMorph scores 99.75% accuracy on known tokens seen in the training data and 46.20% on unknown tokens encountered only in the held-out set. The accuracy that dkrMorph attains in morphological analysis, including lemma and root identification, is 91.92% overall, 98.49% known, and 33.36% unknown.

Whereas morphological disambiguation of a given token selects a full morphological analysis (morphemes and their attributes), the task of discriminative lemmatization disambiguates specifically among the

lemma alternatives by either choosing one or ranking them. Hence, discriminative lemmatization can be seen as a tagging task with an open tag vocabulary consisting of all possible lemmas. For inflections and declensions of word types with regular morphology, a token is likely to resemble a substring of its lemma (e.g., "walked" contains its lemma "walk"). For irregular forms, such correspondence is less likely (e.g., "went" versus "go"). Because of the substring correspondence of most word tokens and their lemmas, discriminative lemmatization can be implemented using general machine learning approaches to the *string transduction* problem, that is, transforming one string into another. The string transducers may be trained from token-lemma pairs or stem-lemma pairs as applicable to the lemmatization formulation. For languages with rich inflectional morphology, lemmas are highly useful for conflating related terms and thereby increasing recall in information retrieval (cf. Ahmed and Nürnberger, 2011). In particular, as part of this work (see Chapter 6) we evaluate several well-performing machine-learned string transduction models, namely Morfette (Chrupała, 2006; Chrupała et al., 2008), a variant of Morfette called hybrid-Morfette (McClanahan et al., 2010), and DirecTL+ (Jiampojamarn et al., 2008).

2.4 Homographic Headword Disambiguation

Returning to the second step in CDL, after identifying the headword (lemma) for the word token, the next step is homographic headword disambiguation to select the best dictionary entry with the selected headword. If a dictionary consistently distinguished multiple entries with the same headword by part of speech or word sense, then homographic headword disambiguation could be accomplished with the tasks of part of speech tagging (cf. Abney, 1997) or word sense disambiguation (cf. Navigli, 2009) respectively (see the the dictionary (lower) portion of Figure 2.1). However, human-crafted dictionaries, including the Payne Smith (1903) Syriac dictionary, are not always consistent in their structure and/or may not have a convenient digital encoding of part of speech or word sense. Lacking consistent distinction by part of speech or word sense, predictive models can still be employed to select the best entry under a headword (lemma) using context features for the current word token. In this work we take this context-sensitive approach to homographic headword disambiguation since it is the most general formulation.

2.5 Dictionary Curation

Related to the CDL task is the task of dictionary curation which is to populate and refine the dictionary. Indeed, CDL is one data-driven aid to dictionary curation which focuses on automatically populating the dictionary with usage examples. Another data-driven tool that lexicographers may employ is the Word Sketch Engine (Kilgarriff and Rundell, 2002; Kilgarriff, 2005). A *word sketch* summarizes the words that co-occur with a particular target word in a corpus, organized by the frequency of the collocations and morphological attributes of co-occurring words. Word sketches offer a lexicographer information with which to manually build and refine a dictionary including part of speech and word sense information, but word sketches do not automatically link word occurrences to existing natural dictionary entries. Thus, word sketches and CDL are complementary, although the same kind of collocation features captured in word sketches may help to improve the quality of automatic CDL.

2.6 Summary

In this work we seek to simulate corpus-dictionary linkage for classical Syriac and Quranic Arabic. We employ the two-step formulation of lemmatization that is well-suited to Semitic languages, with morphological segmentation and stem-to-lemma linkage sub-tasks. Sequence tagging models achieve high accuracy for both sub-tasks but there is greater room for improvement for the stem-to-lemma linkage task (see Section 2.3). We represent dictionary entries with a headword (lemma) and a numeric ID. Thus, the sequence tagging models we develop for homographic headword disambiguation may only use features from the token's local context to decide on a ranking of the candidate dictionary entries with the same headword.

Chapter 3

Related Work

In this chapter we review work related to interactive pre-annotation methods, and sequence tagging models. The previous chapter (Chapter 2) reviews work related to the corpus-dictionary linkage annotation task. Interactive pre-annotation assistance has the potential to accelerate expert annotation of even low-resource languages provided the model is sufficiently accurate and can be updated quickly. Previous work, which we identify in Section 3.1, has not investigated the feasibility of providing interactive and context-sensitive pre-annotation assistance so this thesis makes a first contribution in demonstrating that interactive and context-sensitive pre-anntotation assistance is practical and effective. We employ a sequence tagging formulation of the corpus-dictionary linkage task so that pre-annotations can be context-sensitive, and will therefore use sequence tagging models in our solution. We review sequence tagging and sequence models in more detail in Section 3.2.

3.1 Interactive Pre-annotation Methods

In this thesis we require that machine learning algorithms execute quickly in order for pre-annotation assistance to be deemed interactive. In general, the goal common to all interactive machine learning is to produce a better model with human involvement. How quickly the training and prediction algorithms execute, and thus the time that the human users wait for updated predictions, may or may not be relevant in calling a method "interactive", depending on, respectively, whether the primary goal is to assist the humans to accomplish a task (e.g., Fails and Olsen, 2003) or to build a better model with human feedback (e.g., Druck and McCallum, 2011). Since our ultimate goal is to assist humans by accelerating expert annotation, our use of the term "interactive" includes the requirement that the training and prediction algorithms execute quickly.

Another view on interactive learning is coactive learning. In the coactive learning setting the human

(or humans) and model mutually learn from one another such as in web searches and movie recommendations (Shivaswamy and Joachims, 2012), and machine translation (Sokolov et al., 2015). While human feedback may not always be optimal—complete, correct and consistent—and is most easily expressed as preferences instead of as cardinal utility (numeric scores), coactive learning algorithms can still learn from human preference feedback so as to minimize the regret, or difference, between what is predicted and what the humans prefer. While we do not conduct a regret bounds analysis for CDL using the coactive learning paradigm, we trust that our methods can in fact learn from non-optimal human feedback which is expressed as preferences by making corrections to segmentation, lemmatization, and dictionary headword pre-annotations.

Interactive pre-annotation tools have been built and evaluated for classification tasks but have not been adequately developed and evaluated for context-sensitive tasks. Interactive machine assistance systems have been built for classification tasks such as document categorization (Settles, 2011) and handwritten census record field indexing for small-class fields such as relationship, gender, marital status, and location (Clawson and Barrett, 2015). Providing interactive machine assistance which is context-sensitive is more difficult. For our purposes dealing with text annotation, context-sensitive and interactive pre-annotation assistance has two requirements (Culotta et al., 2006): (1) as soon as an annotator provides a new annotation or correction, **requery** the pre-annotation model for updated suggestions for surrounding words or other parts of a multi-part annotation involving the current token; (2) regularly and quickly **refine** the model by retraining with the expanded and corrected annotated data. The current work develops pre-annotation assistance which satisfies both requirements, unlike previous research which has either not requeried for updated pre-annotations while a sentence is annotated (Haertel et al., 2010a) or has not continually refined the model (Culotta and McCallum, 2005; Culotta et al., 2006).

The prediction method of correction propagation supports the requery requirement for interactive and context-sensitive pre-annotation. **Correction propagation** suggests labels for unannotated tokens in a sequence so as to be consistent with the annotated tokens' labels (Kristjansson et al., 2004). The model does not need to be retrained before correction propagation can offer updated suggestions. Kristjansson et al. (2004) only evaluated the propagation of the first correction per sequence, after which the remainder of the sequence was corrected by the simulated annotator without propagating corrections. Felt et al. (2013) applied correction propagation for all manual corrections, both across the sequence and for multi-part annotations on the current token. They demonstrated that correction propagation for a model that achieves

moderately good held-out accuracy—80% or above in their case—can significantly reduce the time for human annotators to annotate Syriac morphology. This thesis likewise employs correction propagation to support the requery requirement.

Refining predictive models is common in stream processing and active learning applications, although the interest in quick training algorithms varies. Real-time stream processing applications, such as forecasting stock price changes, are concerned with quickly analyzing and learning from a data stream so as to provide up-to-date responses to the current state of the stream (cf. Stonebraker et al., 2005). Active learning uses the machine-learned model to select unannotated data examples for human annotators to annotate, after which the model is retrained to include the new annotations (cf. Settles, 2012; Haertel, 2013). Active learning research usually compares data selection strategies by the number of data items that need to be manually annotated to get the same level of model performance: the smaller the number of manually annotated data items then the better is the selection strategy. Traditional evaluation methods for active learning simply train fresh and independent models on snapshots of increasing amounts of annotated training data, thus the time to retrain the models is seldom of concern since annotation of each data item is not simulated. The active learning work by Haertel et al. (2010a) is a counter example which does retrain the model after simulating annotation of each sequence in an English part of speech tagging task. To validate that our training algorithms are suitably interactive (i.e., quick), we likewise conduct simulations in which the model is retrained after each sentence and additionally we conduct other simulations in which the model is retrained after each token is annotated.

For a pre-annotation system to be interactive, the requery and refine operations need to execute within a few seconds. Thomas and Cook (2005) summarize cognitive science and information visualization literature to emphasize that human perception of interactivity allows for different ranges of execution time depending on the complexity of user-initiated tasks. Simple tasks, such as opening a web page by clicking a hyperlink, should take only a few seconds, whereas more complex user-initiated tasks may reasonably take on the order of ten seconds to execute, provided that some visual indication is given to assure the user that something is being done. Annotators will not want to wait for more than a few seconds after correcting or validating an annotation before they can proceed with their work and still have pre-annotation support.

Executing the refine and requery operations in parallel can facilitate interactive pre-annotation assistance even when the time to train the model is more than a few seconds. A pre-annotation system may execute the refine (train) and requery (predict) steps in sequence, or in parallel by using the old model to

continue to provide pre-annotations while training a new model in another thread or process (Haertel et al., 2010a). Settles (2011) built an interactive text documentation classification system, called DUALIST, which trains and predicts in sequence in less than a second (see also Settles and Zhu, 2012). However, training sequence tagging models usually takes more than a second. Haertel et al. (2010a) simulated English part of speech sequence tagging and trained the model and predicted pre-annotations in parallel. They do not report model training time directly, but the time to train a new model was usually shorter than it took the simulated annotator to submit a corrected sequence and request another pre-annotated sequence to correct. Haertel et al.'s system could reasonably be deemed interactive at a sentence level because the pre-annotation model learns from annotation corrections, and improvements in accuracy can usually be observed in each successive query when the pre-annotations are provided with a newer model which has been trained with a larger amount of data. The linear regression model used to simulate the annotator's time to validate and correct English part of speech sequence pre-annotations comes from Ringger et al. (2008). Haertel et al. (2010a) used the sequence timing model but the roughly equivalent token timing model has a cost of 14–20 seconds per token. While English part of speech tagging is different than corpus-dictionary linkage, and a timing model for CDL does not exist, we anticipate that human annotators will on average take comparable time per token to annotate. Thus, in this work we aim for training times per model update with each newly annotated token that are 10 seconds or less (more details discussed in Chapter 7).

3.2 Sequence Tagging Models

Context-sensitive text annotation is often formulated as a sequence tagging task. Sequence tagging is an instance in the general problem domain of structured prediction. For a given input sequence **x** (e.g., sentence), a sequence tagging model predicts a tag sequence **y** (e.g., dictionary links) by predicting a tag for each token in **x**. Predictions are determined by the features from the token itself and its surrounding tokens and tags and the model's feature weights. State-of-the-art sequence tagging models are trained using discriminative methods which tune feature weights according to an error or loss function calculated using the tag sequence the model currently predicts and the desired tag sequence so that in the future the model will more likely predict the desired tag sequence. We review probabilistic and linear sequence tagging models.

The probabilistic sequence tagging model best suited to adapt for interactive pre-annotation assistance is the maximum entropy Markov model (MEMM). Probabilistic sequence tagging models which are

discriminatively trained score candidate tag sequences for a given input sequence using the conditional probability of the output sequence given the input sequence: $P(\mathbf{y}|\mathbf{x})$. The model is trained by adjusting the feature weights so that the conditional log-likelihood of the data is maximized. Such models include the **maximum entropy Markov model** (MEMM) (McCallum et al., 2000) and the **conditional random field** (CRF) (Lafferty et al., 2001). CRFs are the *de facto* choice for probabilistic, discriminative sequence tagging, since CRFs can model dependencies between tags better than does the MEMM when the tag transitions present in the training data are sparse (cf. Lafferty et al., 2001); however, in practice MEMMs still achieve high accuracy on most real problems when features from the current input token and surrounding tokens are used to inform the prediction of the label at the current position (e.g., Haertel et al., 2010b; McClanahan et al., 2010; Sutton and McCallum, 2012). Training CRF models can take minutes to days in general (cf. Sutton and McCallum, 2006; Tsuboi et al., 2008; Sutton and McCallum, 2012), whereas training MEMMs takes seconds to minutes for our CDL problem (see Section 6.4). We use MEMMs throughout this research because they achieve good accuracy, are quick to train, and are sufficient to demonstrate that sequence taggers can be trained interactively.

Several training algorithms for linear models could be adapted for interactive sequence tagging. Linear models score candidate tag sequences $\hat{\mathbf{y}}$ for a given input sequence $\mathbf{x}$ by using a weighted linear combination of the features: $\mathbf{w}^T\Phi(\mathbf{x},\hat{\mathbf{y}})$, where $\mathbf{w}$ is the vector of feature weights and $\Phi(\mathbf{x},\hat{\mathbf{y}})$ is the vector of features from the input-candidate sequence pair. Many training algorithms adjust the feature weights online, or incrementally, by iterating over the input-output sequence pairs in the training data and for each pair predicting a tag sequence, then calculating the loss function, and finally adjusting the feature weights. The fact that the training algorithms are already incremental by nature means that they have potential to be used interactively, provided the algorithms are adapted to handle partially annotated training sequences. Decoding (predicting) the best tag sequence for a given input sequence is typically done with a variant of the Viterbi algorithm (Viterbi, 1967). Exact decoding has the same computational complexity as calculating the CRF normalizing term, but approximate decoding techniques yield good results with a faster practical run time (e.g., Huang et al., 2012), and the exact decoding algorithm developed by Kaji et al. (2010) runs at speeds comparable to approximate techniques. The loss function and feature weight adjustments can be done in a variety of ways. Collins (2002) adapted the classic **perceptron** and **averaged perceptron** training algorithms for classification to apply to language sequence tagging problems. The perceptron or averaged perceptron training algorithms are easy to execute and tend to result in better accuracy than the MEMM

(e.g., Collins, 2002; Seddah et al., 2010). The **Margin Infused Relaxed Algorithm** (MIRA) (Crammer and Singer, 2003) is a training algorithm that typically achieves better predictive accuracy than the perceptron algorithm for classification (e.g., Crammer and Singer, 2003) and sequence tagging (e.g., Jiampojamarn et al., 2008) but requires solving a quadratic optimization problem which is likely nontrivial, although we are not aware of work that reports empirical training times. Other **Passive-Aggressive** (PA) training algorithms which maintain a less aggressive margin than does MIRA achieve about the same classification accuracy as MIRA but (qualitatively) execute more quickly (Crammer et al., 2006), which is likely true for sequence tagging as well. We do evaluate the use of a linear model trained with MIRA for the stem-to-lemma linkage CDL sub-task in Chapter 6, but we do not employ linear models for the whole CDL pipeline.

Chapter 4

Thesis Statement

Context-sensitive sequence tagging models for corpus-dictionary linkage annotation are made viable for interactive pre-annotation assistance by training the models incrementally on partially annotated data and adding new possible labels and features to the models as they are encountered. Interactive pre-annotation assistance mitigates the need for annotated training data prior to employing pre-annotation assistance to accelerate the development of corpus-dictionary linked data for low-resource languages.

Chapter 5

Evaluation Data Sets

In this chapter we review the Quranic Arabic and classical Syriac data that we use in our experiments to validate methods for interactive pre-annotation assistance for low-resource corpus-dictionary linkage settings. In Section 5.1 we first explain why we evaluate CDL for Quranic Arabic and classical Syriac and justify how the evaluation pertains to low-resource language annotation. We then review the morphology, existing CDL resources, and the details of the Arabic data set we use in Section 5.2 and similarly for Syriac in Section 5.3. In particular, we employ the morphologically annotated Quranic Corpus (Arabic) and Peshitta New Testament corpus (Syriac). In Section 5.4 we describe how we construct artificial dictionaries from the morphologically annotated corpora, how we process the data, and analyze the ambiguity in each data set for each CDL subtask. We finish in Section 5.5 by elaborating on the scope of the Syriac Electronic Corpus project which is the motivation for studying machine assistance for corpus-dictionary linkage.

5.1 Why Quranic Arabic and classical Syriac?

In order to properly assess the utility of our chosen data-driven methods for corpus-dictionary linkage in low-resource settings, we focus on two Semitic languages: Arabic and classical Syriac. Granted, neither Arabic nor Syriac suffer from a paucity of annotated resources. In fact, for each language we rely on a morphologically annotated corpus—not a trivial resource—for our experiments. Nonetheless, we evaluate our methods on these two data sets in simulation as if they were truly representative of low-resource conditions, beginning with little or no initial train with small portions as though starting from zero. In the simulations the annotations are not employed in bulk but in incremental doses, as though acquired from annotators for the first time. Our evaluations in Chapter 6 begin with just one annotated sequence available for training the initial model, and in Chapter 7 the evaluations start with zero annotated sequences. Hence, the term "low resource" better describes the scenario than the data sets employed for the experiments. Therefore, the results

presented in this work shed light on the potential annotation acceleration using interactive pre-annotation assistance for truly low-resource conditions.

We have chosen Arabic because it is an influential Semitic language and is the more highly resourced counterpart to our other choice, classical Syriac. Results in Arabic will reinforce and shed additional light on the results acquired for classical Syriac. Arabic exhibits diglossia: the commonly used and largely standardized written language (Modern Standard Arabic, henceforth MSA) contrasts sharply with numerous spoken dialects, not all mutually intelligible. MSA derives largely from classical Arabic, the language used in the Quran, and so our evaluation on Quranic Arabic is likely representative of MSA corpus-dictionary linkage as well. Analyzing and describing all forms of Arabic language constitutes an active research area in corpus linguistics and natural language processing, with many language resources having been created and disseminated as a result.[1] Even so, most Arabic dialects are low-resource, having few or no annotated language resources.

By text volume, Syriac is the largest dialect of Aramaic and an object of keen interest to our collaborators. Classical Syriac was the primary spoken and written language of the Christian Near East through the eighth century and is still used in the liturgy of the present Syriac Christian churches. Briefly, evaluating our methods on classical Syriac is step towards the larger aim of supplying scholars machine assistance so that they can more quickly link the corpus of Ephrem the Syrian (d. 373 C.E.) to a Syriac dictionary. Since Ephrem is one of the most prolific authors in classical Syriac, linking Ephrem to the dictionary will foster additional corpus studies and scholarship in the corpus.

5.2 Arabic Morphology and Resources

Arabic is a morphologically rich language which makes it a nontrivial task to associate a word token in written text with its dictionary entry. Arabic words, as with other Semitic languages, are comprised of the following parts, with examples given in Table 5.1:

- **stem**: the core part of a word; compound words have multiple stems
- **prefix**: clitic preceding the stem; some words have no prefix while others have one or more prefixes
- **suffix**: clitic following the stem; some words have no suffix while others have one or more suffixes
- **lemma**: (i.e., baseform) the basic word-form from which stems are inflected

[1] See, for example, `https://sites.google.com/site/nlp4arabic/`.

- **root**: the abstract word-form from which lemmas are derived; many Semitic roots, including Arabic's, consist of three literals.

Stems are usually morphological inflections of a lemma and so are self-standing words, yet Arabic dictionaries only define the lemmas and roots of the language (e.g., Lane, 1863). As an analogy in English, a dictionary would contain entries with the headwords "*pen*" and "*jog*" but not for "*pens*" and "*jogging*" which are inflections of the preceding headwords respectively. In this work we link an Arabic word token to a dictionary entry by automatically predicting its stems (morphological segmentation), and then predicting the lemma for each stem (stem-to-lemma linkage), and finally predicting an appropriate dictionary entry for each lemma with the lemma as the entry headword (homographic headword disambiguation). We do not link lemmas to roots although that certainly can be included in the CDL formulation (e.g., McClanahan et al., 2010).

إياك نعبد وإياك نستعين اهدنا الصراط المستقيم

iyyāka na'budu wa-iyyāka nasta'īnu ih'dinā l'ṣirāṭa l-mus'taqīma

It is You we worship and You we ask for help. Guide us to the straight path -

Word Token		Prefix	Stem	Suffix	Lemma
5:1	*<iy~aAka*		*<iy~aAka*		*<iy~aA*
5:2	*naEobudu*		*naEobudu*		*Eabada*
5:3	*wa<iy~aAka*	*wa*	*<iy~aAka*		*<iy~aA*
5:4	*nasotaEiynu*		*nasotaEiynu*		*{sotaEiynu*
6:1	*{hodinaA*		*{hodi*	*naA*	*hadaY*
6:2	*{lS~ira‘Ta*	*{l*	*S~ira‘Ta*		*Sira‘T*
6:3	*{lomusotaqiyma*	*{lo*	*musotaqiyma*		*m~usotaqiym*

Table 5.1: Detail from the Quran chapter 1, verses 5 and 6. First line: Arabic script (right to left). Second line: phonetic transcription (left to right). Third line: English gloss (left to right). Table: orthographic transliteration of each word token and its prefix, stem, suffix, and lemma encoded in the Buckwalter transliteration scheme extended to handle Quranic symbols (Dukes, 2013).

The experiments reported in this work employ morphologically annotated data from the Quranic Corpus (Dukes and Habash, 2010).[2] Arabic, including the majority of MSA text, is generally written without diacritics—vowels and other pronunciation marks—but because the Quran must be accessible to the many Muslims who are non-native Arabic speakers it is fully diacritized. Thus, the Quranic Corpus can be used as two distinct data sets which we denote as QC/D and QC/U for diacritized and undiacritized versions,

[2]The Quranic Corpus website (`http://corpus.quran.com/`) offers an interactive interface for consulting Quranic text, visualizing the morphological and conceptual structure of individual words, and linking them to online lexical descriptions.

respectively, and QC/* to collectively refer to both data sets. Table 5.3 summarizes the QC/D and QC/U data sets, and shows that when diacritics are stripped that there is more ambiguity to the morphological segmentation and stem-to-lemma linkage tasks. We discuss the ambiguity of the homographic headword disambiguation task in more detail in Section 5.4.

5.3 Syriac Morphology and Resources

Syriac, like Arabic, is also a morphologically rich language and thus it is a nontrivial task to link word tokens in written text to their dictionary entries. Classical Syriac has twenty-two letters (literals) like Hebrew and is usually written with very few diacritics for vowels or other pronunciation marks. As explained for Arabic, and as shown in Table 5.2 and Figure 2.1, Syriac word tokens are also comprised of stem, prefix, and suffix parts. Each stem is a morphological inflection of a lemma, which in turn is derived from a root. Syriac dictionaries, such as The Jessie Payne Smith *Compendious Syriac Dictionary* (Payne Smith, 1903), hereafter JPS, focus on indexing and defining lemmas and roots; JPS contains more than 16,000 entries. In this work we link word tokens to stems and stems to lemmas and leave linkage of lemmas to roots to future work.

ܐܡܪ ܠܗܘܢ ܦܝܠܛܘܣ ܠܡܠܟܐ ܐܙܩܘܦ

’āmar lhún pílatāws lmalkkún ’ezqúp

Pilate saith unto them, Shall I crucify your King?

Word Token		Prefix	Stem	Suffix	Lemma
9	*’mr*		*’mr*		*’mr*
10	*lhwn*		*l*	*hwn*	*l*
11	*pylṭws*		*pylṭws*		*pylṭws*
12	*lmlkkwn*	*l*	*mlk*	*kwn*	*mlk’*
13	*’zqwp*		*’zqwp*		*zqp*

Table 5.2: Detail from a portion of John 19:15 in the Peshitta New Testament. First line: Syriac script (right to left). Second line: phonetic transcription (left to right). Third line: English gloss (left to right). Table: orthographic transliteration, excluding diacritics, of each word token and its prefix, stem, suffix, and lemma encoded in the Library of Congress’ transliteration scheme for Syriac (`http://www.loc.gov/catdir/cpso/romanization/syriac.pdf`).

The experiments reported in this work employ morphologically annotated data from the Peshitta New Testament (PNT). The original morphological annotation of the PNT was done manually by The Way International Foundation over the course of fifteen years. Later, the annotated corpus was digitized and

refined into the Syriac Electronic Data Retrieval Archive III (SEDRA3) by Kiraz (1994). Subsequently McClanahan et al. (2010) refined SEDRA3 to include word segmentation annotations that identify the stems and at most one prefix and suffix parts of a word; thus, multiple prefixes and suffixes in a word are grouped as a single unit in the annotation. The tokens and stems in the PNT are fully diacritized, so like the Quranic Corpus, the PNT can be used as two distinct data sets which we denote as PNT/D and PNT/U for diacritized and undiacritized versions, respectively, and as PNT/* to collectively refer to both data sets. SEDRA3 contains information to diacritize the lemmas but in this work we only use undiacritized lemmas in both PNT/D and PNT/U. Table 5.3 summarizes the statistics of the PNT/D and PNT/U data sets. As with the QC/* data sets, there is more ambiguity in the morphological segmentation and stem-to-lemma linkage tasks for the undiacritized PNT data set than for the diacritized one. We discuss the ambiguity of the homographic headword disambiguation task in more detail in Section 5.4.

	Arabic		**Syriac**	
All	**QC/D**	**QC/U**	**PNT/D**	**PNT/U**
verses	6236		7957	
tokens	77,429[a]		109,640	
tokens/verse	12.42		13.78	
word types	18,993	14,880	18,359	16,439
prefix types	76	74	79	19
stem types	12,134	7469	11,386	7970
suffix types	234	95	104	51
lemma types	4911	4448	3038[b]	
stem POS types	54		8	
tokens w/ 2+ stems	49	2668	146	2930
stem instances w/ 2+ lemmas	4639	11,334	2350	17,716
lemma instances w/ 2+ POS	18,367	36,986	15,232	
compound tokens	486		282	
mixed compound tokens	666	1057	0	2

Table 5.3: Statistics for the four data sets. Notes: (a) The number of tokens in the QC/* data sets is one less than reported by Dukes and Habash (2010). (b) The PNT/* data sets both use undiacritized lemmas.

5.4 Dictionary and Corpus Processing and Analysis

To simulate CDL annotation projects we employ morphologically annotated corpora and create artificial dictionaries from the lemma and part of speech annotations. We are not aware of any annotated data sets that link corpus word tokens and natural dictionary entries, thus we instead create artificial dictionaries using

lemma and part of speech (POS) annotations from morphologically annotated data sets. The lemmas will be used as the headwords of the artificial dictionary entries, and the POS tags will be used by the simulated annotator to determine when to create distinct (by numeric ID) dictionary entries with the same headword. The POS tag will not be available as a feature to the automatic annotation models. Given this construction, the pre-annotation models will only have access to corpus text and the dictionary headwords from which to derive features in order to predict links, whereas if natural dictionary entries were available then the text of the entries could also be used as a source of features.

We made some simplifying assumptions in processing the data sets for use in our simulations. Annotated language data on the scale of tens of thousands of tokens is not perfectly consistent and captures interesting edge cases in the language but which are not common. In the Quranic Corpus data set, not all of the tokens had a lemma annotation so we used the text of the stem for the lemma. As reported in Table 5.3, both the QC/* and PNT/* data sets contain compound tokens which contain two or more stems, each potentially with its own lemma (the PNT/* is missing some of the lemma annotations). Sometimes a compound word type's instances are not annotated uniformly: some instances are annotated as a compound and some are not, and sometimes different instances have different compound annotations; these are reported as mixed compounds in Table 5.3. The volume of compound tokens and mixed compound tokens as percentage shares of the corpora is very small: compound tokens make up 0.6% of QC/* and 0.2% of PNT/*, mixed compound tokens make up 0.8% of QC/D and 1.4% of QC/U and almost no portion of PNT/*. We handle compound tokens by simply concatenating the multiple stems together to form a single "stem". Similarly, we concatenate multiple prefixes, suffixes, and lemmas for a token into a single prefix, suffix, and lemma respectively. Because the number of compounds is small then our approach to concatenate the parts will not dramatically affect the generalization ability of the pre-annotation models.

The ambiguity in the data sets for each of the different CDL sub-tasks demonstrates the difficulty of each sub-task. Table 5.3 reports relevant statistics for each data set. When the data is more ambiguous there is more need for context-sensitive disambiguation in order to provide good pre-annotations. The sub-tasks get progressively more difficult, that is they have more need for context-sensitive disambiguation. The difficulty of the morphological segmentation sub-task is seen in the number of tokens with two or more stems: the undiacritized data sets contain 20 to 54 times more such tokens than do the diacritized data sets. The difficulty of the stem-to-lemma linkage sub-task is seen in the number of stem instances with two or more lemmas: the undiacritized data sets contain 2 to 8 times more such stem instances than do the

diacritized data sets. The difficulty of the homographic headword disambiguation sub-task is seen in the number of lemma instances with two or more entries, backed by part of speech tags: the QC/U data set has twice as many such lemmas as the QC/D data set, whereas the PNT/D and PNT/U data set have the same lemma ambiguity since they use the same undiacritized lemmas. Generally, each sub-task is more difficult for the undiacritized data sets than for the diacritized ones because tokens, stem instances, and lemma instances which are orthographically distinct when diacritized become conflated when the diacritics are stripped.

5.5 The Syriac Electronic Corpus Project

While this work focuses on developing interactive, CDL pre-annotation assistance that can handle corpora with tens of thousands of tokens, the long-term goal of our research group is to develop interactive pre-annotation assistance that scales to millions of word tokens. To appreciate the scale of the data to which the CDL models will ultimately be applied, consider the Syriac Electronic Corpus (SEC) project of the Center for the Preservation of Ancient Religious Texts in the Neal A. Maxwell Institute for Religious Scholarship at Brigham Young University. The aims of the project are to collect, digitize, annotate, and publish a large corpus of classical Syriac texts spanning multiple authors and centuries. To date the project has collected and digitized almost six million words of text, including a sub-corpus consisting of all of the writings of Ephrem the Syrian (d. 373 C.E.). This sub-corpus, subsequently referred to as EPHREM, is of particular historical interest since Ephrem's writings reveal much about early Christian theology. The next step in the SEC project is to link each word in EPHREM to entries in the JPS dictionary and to extend the dictionary where entries are missing.[3]

Applying the interactive, CDL pre-annotation assistance methods developed in this work to EPHREM would test the ability to scale to hundreds of thousands of tokens. EPHREM contains approximately 465,000 word tokens, more than four times as many as the Peshitta New Testament. By linearly interpolating from annotation timing data collected by Felt et al. (2013), in their user study of machine assistance for Syriac morphological annotation, we estimate it might take a single typical Syriac scholar more than two years to fully manually link EPHREM to the JPS dictionary—assuming the annotator works 20 hours per week (48 weeks per year) and does not need to add or correct dictionary entries. The time to fully annotate is reduced

[3] http://cpart.maxwellinstitute.byu.edu/home/sec/about

to under one year if automatic pre-annotations are provided and automatically refined as the annotator corrects the suggested annotations.[4] Thus, machine assistance has the potential to play a critical role in this particular corpus-dictionary linkage project, and making the pre-annotation assistance interactive should further reduce the time to annotate. Beyond EPHREM, machine assistance is even more vital for annotating the entire SEC.

An important benefit of CDL is that the dictionary is improved over time with the addition of new dictionary entries and usage examples for existing and new entries. Linking EPHREM to the JPS dictionary will necessarily require the addition of new dictionary entries. We estimate the number of unique stem and lemma types to be identified in the linkage from EPHREM to the JPS dictionary by linearly interpolating from corpus metrics on the PNT. Assuming EPHREM displays the same token-to-type ratio as the PNT, EPHREM will contain approximately 70,000 word types, 34,000 stem types, and 13,000 lemma types. Although the JPS dictionary contains over 16,000 distinct entries for roots and lemmas, it will undoubtedly need to be expanded in order to accommodate new types as EPHREM is linked. When the entire SEC is linked we anticipate that the JPS will need to be expanded substantially.

[4]These time estimates are lower bounds since EPHREM contains more poetic and metaphorical language than the New Testament apocrypha text studied by Felt et al. (2013).

Chapter 6

Stem-to-lemma Linkage Model Comparison

In this chapter we focus on evaluating methods for stem-to-lemma linkage inasmuch as that is the most difficult and critical subtask to the corpus-dictionary linkage task. Our goal is to determine which stem-to-lemma linkage method is the most promising to adapt for interactive pre-annotation assistance, which we will then use in the next chapter in our solution for complete CDL annotation. Throughout the analyses and evaluations in this chapter we use gold-standard stem-lemma sequence pair data from the morphologically annotated QC/* and PNT/* data sets (see Chapter 5). We begin by describing the data sets in our experiments in Section 6.1. Then, in Section 6.2, we analyze the expected ambiguity of the stem-to-lemma linkage task through the course of a CDL annotation project. We next present four machine learning string transduction models which we employ for stem-to-lemma linkage in Section 6.3. We evaluate the accuracy and training time of the four models on the stem-to-lemma linkage data from the QC/* and PNT/* data sets in Section 6.4. Finally, in Section 6.5, we conclude that the hybrid-Morfette model is the best suited to adapt for interactive pre-annotation assistance.

6.1 Experiment Design

In this section we discuss how we use the data sets to analyze ambiguity during the course of annotation and evaluate model performance. As mentioned in the introduction to this chapter, we use gold-standard stem-lemma sequence pair data from the morphologically annotated QC/* and PNT/* data sets. In our ambiguity analysis we take 80% of each dataset to conduct cross-validated annotation simulations. To develop and evaluate the models we use the same 80% of the data sets as development data and hold out the remaining 20% of the data sets as blind test data which is never used to train nor even select the models in any of the experiments. Table 6.1 reports the number of verses and tokens in the development set and blind test set. A few word tokens in Arabic and Syriac are compounds and have two stems and therefore two lemmas, but

we treat such compounds as having one stem and one lemma by concatenating the parts (see Section 5.4).

	All			**Development** (80% of All)			**Blind Test** (20% of All)		
Source Data	verses	instances	types	verses	instances	types	verses	instances	types
Arabic QC/D	6236	77,429	12134	4988	61,957	10791	1248	15,472	4569
Arabic QC/U			7469			6746			3176
Syriac PNT/D	7957	109,640	11386	6365	87,560	10212	1592	22,080	4802
Syriac PNT/U			7970			7242			3718

Table 6.1: All, Development, and Blind Test data set partitions used throughout this chapter to analyze ambiguity, develop models, and evaluate model accuracy for stem-to-lemma linkage. The columns labeled "instances" and "types" are stem instances and stem types. The numbers for "All" are repeated from Table 5.3 for convenience.

6.2 Ambiguity Analysis

In this section we analyze the expected ambiguity of the stem-to-lemma linkage task through the course of a CDL annotation project. Section 5.4 and Table 5.3 from the previous chapter look at the overall ambiguity of each data set, but in an annotation project the ambiguity distribution will change over time as new annotations are added. We conduct cross-validated annotation simulations using the development data sets (see Table 6.1). The following ambiguity analysis helps us to anticipate how different model mechanisms discussed in the next section might contribute to the overall model accuracy.

The annotated data used to train a model determines what the model knows about the data items for which it will predict pre-annotations. Here we define terminology in terms of stem types and stem instances, but the terminology applies to word and lemma types and instances as well. The set of tags, or labels, associated with a stem type consists of the tags used to annotate stem instances of that type. The **ambiguity** of a word type is the size of its tagset observed in training data. Observed tagset sizes are bucketed as follows:

- An **unknown** stem type has ambiguity 0 with no tags.
- A **known** stem type has one or more tags.
- A **known unambiguous** stem type has ambiguity 1 with exactly one tag.
- A **known ambiguous** stem type has two or more tags; finer-grained partitions may also be used such as ambiguity 2, ambiguity 3, and so on.

Stem types and instances of the same ambiguity can be grouped, resulting in data subsets characterized by

ambiguity. We analyze test share distributions and prediction quality for different ambiguity subsets as well as over all of the data.

To analyze the ambiguity distribution during the course of an annotation project, we measure the ambiguity of test data with respect to increasing percentages (snapshots) of all of the available training data. More concretely, we divide the development data sets into training and development test sets (not the blind test set) and measure the ambiguity of the stem instances in the test set with respect to the lemmas (tags) observed with the stem type in a given percentage of all of the available training data. We sweep the size of the percentage of all of the available training data in order to plot a curve tracing the progress of simulated annotation. The computations are done five times with five-fold cross validation on the development data in order to establish min and max "error" bars for the curves. Figure 6.1 shows the resultant trends for the unknown, known unambiguous and known ambiguous cases in each of the four data sets. We decided to experiment heavily in the region between 0.1% to 20% of the training data to focus on scenarios in which we consider the annotation task to truly qualify as low-resource in terms of the number of annotated tokens and verses: at 0.1% of the training data there are roughly 4 and 5 annotated verses for QC/* and PNT/* respectively, and at 20% there are roughly 750 and 970 annotated verses for QC/* and PNT/*, respectively.

A distinguishing factor between stem-to-lemma linkage models is their ability to predict lemmas for unknown stems, inasmuch as the percentage share of the data that is unknown is very large initially and towards the end of an annotation project still accounts 5% to 10% of stems. Figure 6.1 shows in each of the four data sets that the unknown shares are initially between 77% (PNT/U) and 95% (QC/D) at 0.1% of the available training data and then monotonically decreases—because the test set is fixed—as more of the available data is used for training. The unknown curve finishes off around 5% to 10% of the test data when 100% of the available training data is used. The share of unknown stems constitutes a majority of the test data until about 0.7% (PNT/U) to 2% (QC/D) of the available training data is used for training, after which the known unambiguous stems make up a majority of the test data. Thus, given small amounts of training data, a model's overall prediction accuracy is mostly determined by its ability to predict lemmas for unknown stems. In contrast given larger amounts of training data, a model's overall prediction accuracy is mostly determined by its ability to effectively "memorize" the most frequent lemma for a stem because most stems remain unambiguous once lemmatized. However, a model's ability to handle unknown stems remains important throughout the course of the annotation project, since even with large amounts of training data, 5 to 10 out of every 100 unannotated stems will be unknown. The cross-over point between unknown and

known unambiguous cases occurs earlier for the undiacritized data sets and later for the diacritized ones, an expected result since the undiacritized data sets have fewer stem types than the diacritized sets.

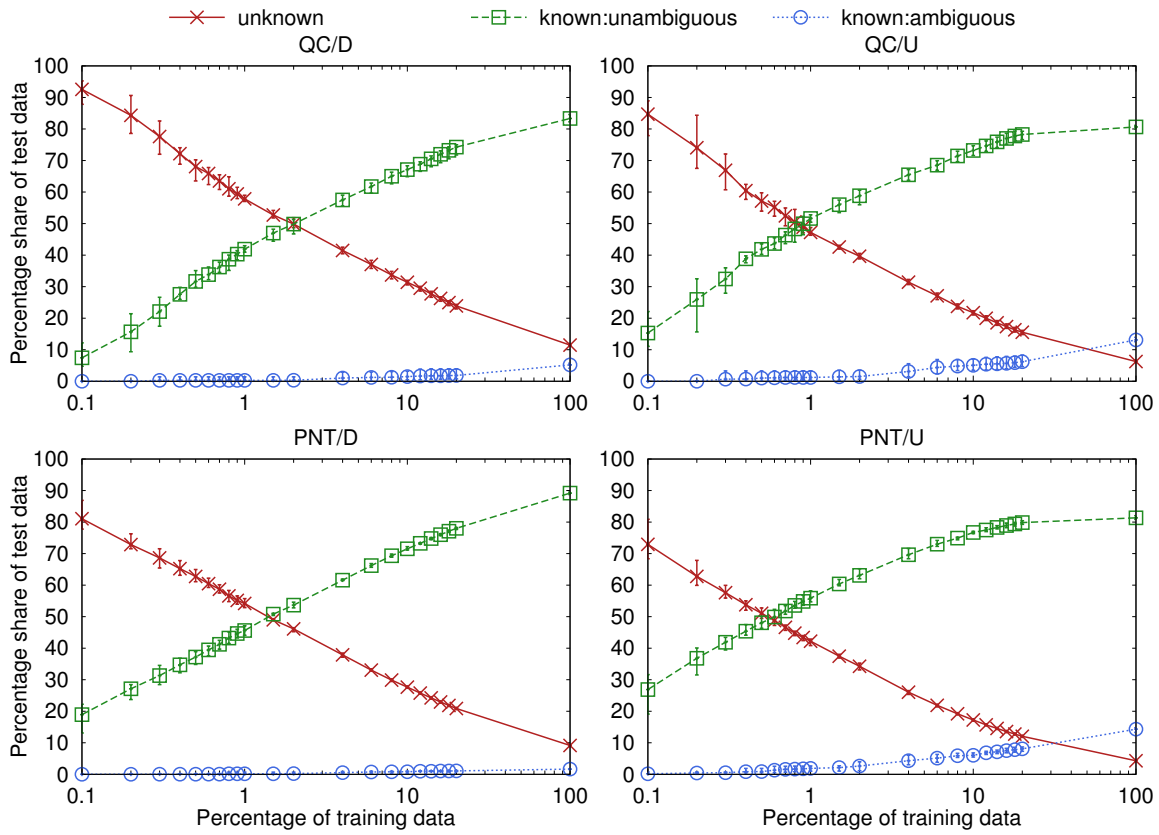


Figure 6.1: Data set ambiguity over the course of annotation. Average percentage shares of the test data comprised of unknown, known unambiguous, and known ambiguous stems. Error bars show min and max percentage shares.

Another distinguishing factor between models on some data sets is their ability to disambiguate among multiple known lemmas for the same stem type. In the trend plots, the share of known ambiguous cases monotonically increases given more training data for all data sets. For the undiacritized data sets (QC/U and PNT/U) the stems become more ambiguous more quickly than for the diacritized sets (QC/D and PNT/D) which is because stem types which were distinct with diacritics become conflated when the diacritics are stripped. Even when there is little training data (0.1% to 20% range), there is enough ambiguity in the undiacritized data sets (tending towards 10%) such that the ability of one model to disambiguate

lemmas better than another may be a distinguishing factor to determine which model does best. For long-term projects, the ability of a model to disambiguate becomes even more critical as seen in the fact that the share of known ambiguous stems surpasses the share of unknown stems around 40% to 50% of the available training data for QC/U and PNT/U.

6.3 Models

In this section we provide more details about the string transduction models which we apply to the stem-to-lemma linkage annotation task. For each model, except the baseline, we experiment with two feature sets: a word-context feature set which only uses features from the current stem to make a lemma prediction, and a local-context feature set which, in addition to the word-context features, also includes features from surrounding stem instances and their lemma annotations. Local context is needed in order for a model to have the capacity to disambiguate among possible lemmas for a given stem. We first describe the baseline model which does not employ local context, and then discuss three more sophisticated models which do use local context.

6.3.1 Baseline Memorizer

The baseline employs simple and easy to train methods for predicting lemmas, but does not have the capacity to disambiguate. For unknown stems, the baseline model uses a heuristic to simply predict the stem itself as the lemma. For known stems, the baseline model memorizes how often each lemma is used with each stem in the training data and then at prediction time predicts the most frequent lemma for the stem type. Because the baseline model does not employ local-context information it cannot disambiguate among the multiple lemmas associated with ambiguous stems. Yet, the baseline model will achieve high accuracy over the course of the annotation project because most stems are known unambiguous (see Figure 6.1).

6.3.2 Morfette

Morfette is a machine learning approach to string transduction that has been shown to perform well on discriminative lemmatization for a variety of languages (Chrupała, 2006; Chrupała et al., 2008). The distinguishing element of Morfette is that its tag set consists of edit scripts to transform stems into lemmas. The edit scripts are deterministically derived from stem-lemma pairs in training data, and they can be applied

to unknown stems having a similar orthographic form as known stems. The method can generalize beyond the lemmas seen in the training data, in so far as the edit scripts can map an unknown stem to a candidate lemma not observed in the training data.

Morfette Mechanics

Morfette predicts a sequence of tags where each tag is an edit script which transforms stems into lemmas. For example, in English, to encode the "DELETE-*'ing'*" transformation, learned from stem-lemma[1] pair "*feeling-feel*", a possible edit script representation using Chrupała's notation is <(DELETE, "*i*", 5), (DELETE, "*n*", 6), (DELETE, "*g*", 7)> (one-based index; cf. Chrupała, 2006); equivalently, we employ a substitution regular expression "/(.4)ing/\1/". The edit scripts are deterministically derived from stem-lemma pairs in the training data using a minimum string edit distance algorithm (cf. Ristad and Yianilos, 1998). Specifically we use the Needleman-Wunsch dynamic programming algorithm (Needleman and Wunsch, 1970)[2] to compute the shortest sequence of insertions and deletions to transform one sequence into another. Lemmas can be derived from predicted edit script tags by applying the script to the stem.[3] Edit scripts derived from one observed stem-lemma pair are applicable to, or generalize to, other word types which match the condition of the edit script; for example, the edit script which encodes the transformation "DELETE-*'ing'*" derived from "*feeling-feel*" can be applied to other word types that end with "*ing*" such as "*writing*" to get a (possibly novel) candidate lemma "*writ*". The original Morfette implementation used a maximum entropy Markov model (MEMM) with *maximum a posteriori* (MAP) training but has since changed to use a weighted linear model and averaged perceptron training.[4] We use the Morfette model implemented in the SyroMorph tool (McClanahan et al., 2010). Our implementation, like SyroMorph, uses standard MAP training for the MEMM but updates SyroMorph to handle Arabic and diacritized Syriac in addition to undiacritized Syriac.

The way in which edit scripts are represented, even for the same set of insert and delete operations, impacts how well Morfette performs. The original Morfette edit script representation (Chrupała, 2006) inserts and deletes specific characters by absolute position from the end of the string. This absolute-end-anchored representation is beneficial for languages with predominantly suffixal morphology (like English), where morphological inflection mostly occurs in the end of the word, because the total number of edit

[1] For the purposes of illustration we define an English word's "stem" to be its raw token form.

[2] The same kind of approach was used by Levenshtein (1966) to handle binary codes.

[3] For the PNT/D data set, edit scripts are derived from stem-lemma pairs where the stem is diacritized but the lemma is not such that the edit script removes all of the diacritics in addition to transforming the literals.

[4] https://sites.google.com/site/morfetteweb

scripts in the tag inventory is reduced when compared to a representation that anchors insert and delete operations from the beginning of the string. For example, with anchoring from the end of the string, the edit script representations for the transformation "DELETE-*'ing'*" are the same for stem-lemma pairs of different lengths, such as "*going-go*", "*trying-try*", "*feeling-feel*", and so on. If operations were anchored from the beginning of the string, then a different edit script would be required for different length tokens that end with "*ing*". Possible ways to generalize edit script representation to further reduce the total size of the tag inventory (hence the number of model parameters) are: (i) to deterministically generalize positioning by anchoring insertion and deletion operations relative to substring patterns; and (ii) to generalize match, insert, and delete operations to operate with character classes, such as consonants or vowels, similar to the root-and-pattern expressions used for classical Syriac (Lindgren, 2011) and for Arabic (Darwish et al., 2014). The representation used in our implementation inserts and deletes characters by absolute position from the beginning of the string; we use this representation in the experiments because it produces the best results compared to different generalizations we have tried. There are also other ways to deterministically derive edits scripts, such as finding the longest common substring and then building an edit tree to describe the transformation which effectively anchors edit operations relative to the start and the end of the word token (Chrupała, 2008). We have not experimented with the edit tree scripts, but Chrupała achieved Polish lemmatization accuracy gains of 0.15 percentage points overall and 0.54 for unknown words, so employing edit tree scripts is promising feature engineering for future work.

Morfette Features

We provide Morfette with the word context and local context feature templates summarized in Table 6.2. All of the feature templates are employed to collectively extract a sparse feature vector for the current stem instance. We denote the models for word context and local context stem-to-lemma linkage as Morfette/W and Morfette/L respectively. Employing only word context features functionally reduces the sequence labeling task to a set of classification tasks for each stem in the sequence.

Word context feature templates include WORD, WORDLEN, CHARACTER, PREFIX, and SUFFIX. The WORD template extracts the stem itself. The WORDLEN template extracts the length of the stem. From the time McClanahan et al. (2010) reported results, we have corrected the CHARACTER template to extract each individual character in the stem; the template had been miscoded to emit *word.length* (length of the stem) number of copies of the same feature, thus being redundant with the WORDLEN feature. The PREFIX

Type	Name	Template	Example
word	WORD	*word*	*'zqwp*
word	WORDLEN	*word.length*	5
word	CHARACTER	*character*	{*'*, *z*, *q*, *w*, *p*}
word	PREFIX	(*length*, *prefix string*)	{(1, *'*), (2, *'z*), (3, *'zq*), (4, *'zqw*)}
word	SUFFIX	(*length*, *suffix string*)	{(1, *p*), (2, *wp*), (3, *qwp*), (4, *zqwp*)}
local	ENDOFSEQUENCE	*distance from end*	N/A[†]
local	PREVTAGS	(*length*, *tag list*)	{(1, *append-'*), (2, *append-'*, *stem-as-lemma*)}[‡]

Table 6.2: Morfette feature templates provided in our implementation. Here *word* means stem. Examples are for stem *'zqwp* in the context of John 19:15 (see Table 5.2) assuming Markov order 2. Note that throughout this example the apostrophe (') character denotes the Syriac letter "alaf". (†) The ENDOFSEQUENCE feature only fires when the word is within a Markov order window of the end of the sentence; *'zqwp* is the 10th stem from the end of the verse so the ENDOFSEQUENCE feature is not extracted. (‡) Using a substitution edit script representation, the *append-'* transformation is */(.*)/\1'/* and the *stem-as-lemma* transformation is */(.*)/\1/*

template extracts the greedy prefixes of length $[1, min(5, word.length)]$ for stems longer than one character. Similarly, the SUFFIX template extracts the greedy suffixes of length $[1, min(10, word.length)]$ for stems longer than one character. Note that the PREFIX and SUFFIX feature templates should not be confused with the prefix and suffix morphemes of a word token: rather, the feature templates extract beginning and ending character subsequences from the stem.

Local context feature templates include ENDOFSEQUENCE and PREVTAGS as well as all of the word context feature templates. The ENDOFSEQUENCE template extracts the distance of the stem from the end of the sequence if the distance is less than the configured Markov order. The PREVTAGS template extracts sequences of lemmas from the stems preceding the current stem; extracted sequences are of length 1 to the configured Markov order (inclusive).

Morfette Parameters

Beyond the hard-coded feature template parameters, our implementation provides other parameters dealing with feature selection, training, and decoding. We use a Markov order of 2. We use all extracted features to train Morfette instead of selecting only rare or non-rare features. We allow the log posterior maximization of the maximum entropy model to run until improvements are less than 10^{-6} or for 250 iterations, whichever

comes first. We use a standard deviation of 1 for regularizing the feature weights. We use only one thread when calculating the log-posterior so as to make the timing results more fair to compare with DirecTL+ which is single threaded. The decoder uses a state beam of size 5 and an arc—or transition—beam of size 5 which are default decoder parameters that provide good predictive accuracy (McClanahan, 2010).

6.3.3 Hybrid-Morfette

SyroMorph's model for stem-to-lemma linkage uses a model composed of various component models, specialized to handle unknown, known unambiguous, and known ambiguous stems, respectively. The component model to handle unknown stems is Morfette: hence the name hybrid-Morfette. As reported by McClanahan et al. (2010), hybrid-Morfette in isolation achieves 96.19% accuracy overall, 98.05% accuracy on known stems, and 78.40% accuracy on unknown stems in ten-fold cross-validation on 90% of the whole PNT/U.

Hybrid-Morfette Mechanics

The hybrid-Morfette model uses different component models to handle stems which are unknown, known unambiguous, and known ambiguous. To handle unknown stems hybrid-Morfette uses a Morfette model trained on all of the training data; this Morfette component in the hybrid has the same word context and local context feature sets as described before. To handle known unambiguous stems hybrid-Morfette predicts the single known, "memorized" lemma for the stem. Hence hybrid-Morfette will perform the same as the baseline memorizer model for the known unambiguous stems, achieving high accuracy over the course of the annotation project (see Figure 6.1). To handle known ambiguous stems, hybrid-Morfette uses a maxent model per stem type trained on instances of the stem to disambiguate among the lemmas known for the stem type.

Hybrid-Morfette Features

For the maxent models that handle known ambiguous stems in hybrid-Morfette, our implementation provides the feature templates summarized in Table 6.3. All of the templates extract features from the local context. The OFFSETPREFIX template extracts greedy prefixes of length $[1, min(4, word.length)]$ from preceding *lemmas* within a maximum window of 3 (negative offsets) and from the current stem and succeeding

stems within a maximum window of 3 (zero and positive offsets). Similarly, the OFFSETSUFFIX template extracts greedy suffixes of length $[1, min(4, word.length)]$ from preceding *lemmas* within a maximum window of 3 and from the current stem and succeeding stems within a maximum window of 3. The preceding lemmas can be used to derive features because the sequence is decoded in sequential reading order. The STARTOFSEQUENCE and ENDOFSEQUENCE templates extract the distance of the stem from the beginning and end of the sequence respectively if the distance is less than 3. The NEARBEG and NEAREND templates fire when the STARTOFSEQUENCE and ENDOFSEQUENCE templates do respectively. The STARTOFSEQUENCE and ENDOFSEQUENCE templates provide a quantitative measurement of the stem from the beginning or end of the sentence, whereas the NEARBEG and NEAREND templates provide a qualitative indication of whether the stem is close to one of the ends.

Name	**Template**	**Example**
OFFSETPREFIX	(±*offset*, *prefix string*)	{ (-2, <), …, (-2, <*iy*~), (-1, *E*), …, (-1, *Eaba*), (0, <), …, (0, <*iy*~), (+1, *n*, …, (+1, *naso*) }
OFFSETSUFFIX	(±*offset*, *suffix string*)	{ (-2, *A*), …, (-2, *y*~*aA*), (-1, *a*), …, (-1, *bada*), (0, *a*), …, (0, *aAka*), (+1, *u*, …, (+1, *iynu*) }
STARTOFSEQUENCE	*distance from start*	2
NEARBEG	*1 or 0*	1
ENDOFSEQUENCE	*distance from end*	1
NEAREND	*1 or 0*	1

Table 6.3: Hybrid-Morfette feature templates provided in our implementation. All templates extract local context features. Examples are for stem 1:5:3 <*iy*~*aAka* in Chapter 1 of the Quran (see Table 5.1). The examples use a maximum offset spans of length 3 and maximum prefix and suffix lengths of 4. Note that the prefixes and suffixes with negative offset from the current stem are derived from the lemmas whereas those with zero or positive offsets are derived from the stems

We denote whether word context or local context feature are used in the Morfette component model that handles unknown stems by Hybrid-Morfette/W and Hybrid-Morfette/L. We did not realize that when training instances of Hybrid-Morfette/W that the OFFSETPREFIX and OFFSETSUFFIX feature templates in the models handling known ambiguous stem types were not constrained to use only word context features until after the experiments were done, thus the maxent component that handles known ambiguous stems uses local context in Hybrid-Morfette/W as well as in Hybrid-Morfette/L. The word context and local context

distinctions thus apply only to the Morfette component model that handles unknown stems.

Hybrid-Morfette Parameters

We used the same parameters for feature selection, training, and decoding for hybrid-Morfette as for Morfette.

6.3.4 DirecTL+

DirecTL+ is a general discriminative string transducer which we apply to perform context-sensitive stem-to-lemma linkage. In essence, DirecTL+ is an application at the character level of modern phrase-based machine translation methods with monotone (non-crossing) alignments. DirecTL+ has been applied to the letter-to-phoneme problem (Jiampojamarn et al., 2008) and name transliteration (Jiampojamarn et al., 2009). Jiampojamarn et al. (2007) recognized the potential to apply DirecTL+ for lemmatization, which Toutanova and Cherry (2009) realized by applying their implementation of the same model for non-context-sensitive lemmatization of English and Czech. To our knowledge, our published work is the first to apply DirecTL+ for *context-sensitive* lemmatization (Black et al., 2014). Learning alignments between token and lemma character sub-sequences in the manner of DirecTL+ has at least one potential advantage over the Morfette method: insert and delete operations are decoupled from one another, whereas in Morfette they are packaged together into a single edit script.

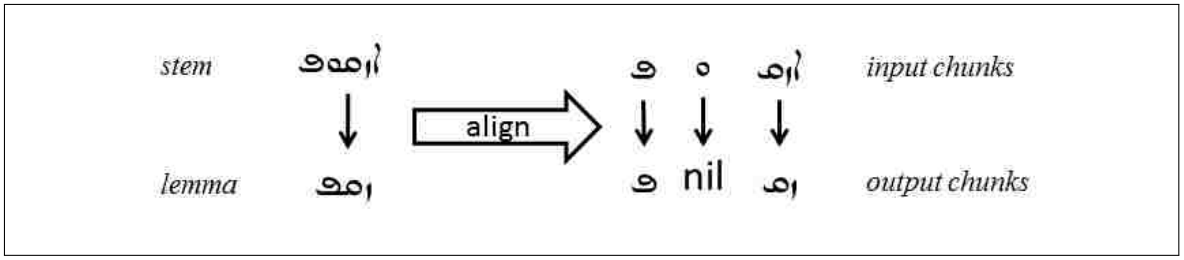
DirecTL+ Mechanics

Aligned character sub-sequences are required to train the DirecTL+ string transducer. DirecTL+ employs a joint structured perceptron and monotone phrasal decoder (Jiampojamarn et al., 2008, 2010); Jiampojamarn et al. (2008) calls the structured perceptron a perceptron hidden Markov model (PHMM), thus we use that terminology even though the model is discriminative, not generative. DirecTL+ is trained from aligned character sub-sequences. Since many string transduction problems do not naturally have training data encoded as aligned character sub-sequences, including the stem-to-lemma linkage data we evaluate, Jiampojamarn et al. (2007) implemented a monotone (no crossing) many-to-many (M2M) alignment algorithm that jointly splits the input and output strings into character sub-sequence "chunks" and aligns the chunks. The M2M algorithm has four key parameters:

- **maxX**: maximum number of characters to allow in an input chunk
- **maxY**: maximum number of characters to allow in an output chunk
- **delX**: allow deletion of input chunks; the input chunk may be aligned to the null or empty output chunk, effectively deleting the input chunk during transduction
- **delY**: allow deletion of output chunks; the null or empty input chunk may be aligned to the output chunk, effectively inserting the output chunk during transduction.

The best M2M parameters are maxX = 3, maxY = 3, delX = yes, delY = no; parameter selection will be explained later. For example, after M2M processing, the undiacritized Syriac stem-lemma pair (*'zqwp*, *zqp*) is encoded with chunks as shown in Figure 6.2a. The PHMM (Collins, 2002) implemented in DirecTL+ uses the Viterbi algorithm, or a beam search adaptation of the Viterbi algorithm, to transduce (predict) the best output string for a given input string. The monotone phrasal decoder works in conjunction with the PHMM to group individual input characters into sub-sequences consistent with the input chunks in the M2M-aligned training data. Thus, at prediction time the input strings to be transduced need not be pre-split into chunks.

(a) Many-to-many alignment of one word



(b) Many-to-many word alignments concatenated

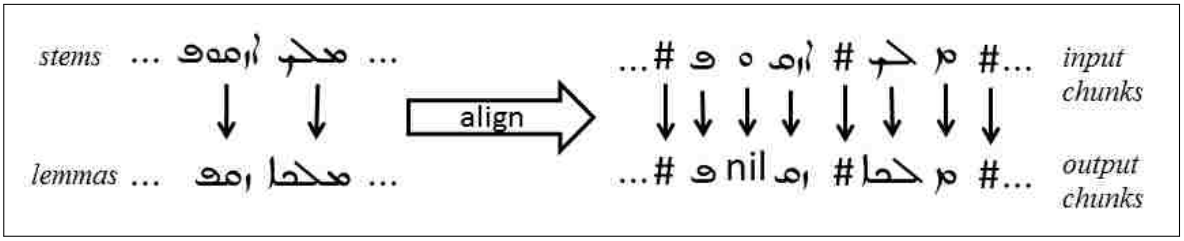


Figure 6.2: Many-to-many alignment examples. Figure 6.2a shows a possible chunking and alignment for the Syriac stem-lemma pair (*'zqwp*, *zqp*). Figure 6.2b shows how the chunkings and alignments of individual stem-lemma pairs can be concatenated together to form an aligned stem-lemma sequence pair for a portion of John 19:15. The hash symbol ("#") is a unique delimiter which marks token boundaries. An input chunk which aligns with "nil" indicates that the input chunk is deleted.

We apply DirecTL+ for word context (non-context-sensitive) and local context (context-sensitive) stem-to-lemma linkage by encoding sequence (verse) data into sets of different kinds of input-output string pairs. We denote the models for word context and local context stem-to-lemma linkage as DirecTL+/W and DirecTL+/L respectively. The M2M utility is first applied to all of the stem-lemma pairs in the training data to acquire aligned training data. For DirecTL+/W, the M2M-aligned training data is used directly to train DirecTL+/W. For testing, the raw input stems are left as they are. For DirecTL+/L, the alignments for the stem-lemma pairs in the M2M-aligned data are concatenated to produce a pair of input and output strings for the sequence (verse). The input string thus encodes the chunked stem sequence, and the output string encodes the chunked lemma sequence. When the alignments are concatenated a unique delimiter chunk is used to demarcate the boundaries between stems and (separately) between lemmas. For example, Figure 6.2b shows how individual stem-lemma pair alignments can be concatenated together to form an aligned stem-lemma pair sequence using a unique token delimiter chunk ("#"). For testing, the raw stems in the sequence are concatenated together using the same delimiter chunk. Because the item delimiter chunk ("#") is always transduced to itself, the concatenating pre-processing method ensures that DirecTL+/L will always predict a lemma sequence with exactly one lemma for each stem in the input stem sequence.[5]

DirecTL+ Features

The original DirecTL+ implementation provides the features summarized in Table 6.4, which can be seen as word or local context features given the particular pre-processing of the data. Each of the templates operates with respect to the current input chunk being transduced. The n-gram template extracts all input chunk n-grams of size $[1, n_{standard}]$ within a window c of the current chunk. The transition template extracts the preceding output chunk. The linear chain template pairs the preceding output chunk with each of the n-grams extracted by the n-gram feature template. The joint n-gram template extracts all n-grams of size $[1, n_{joint}]$ of input-output-chunk pairs preceding the current chunk. With these features and the monotone phrasal decoder that explores possible chunks, DirecTL+ captures much of the same information as the WORD, CHARACTER, PREFIX, and SUFFIX feature templates in Morfette and OFFSETPREFIX and OFFSETSUFFIX feature templates in hybrid-Morfette. DirecTL+ does not provide equivalent feature templates

[5] An alternative pre-processing method of forming stem and lemma sequence strings interposed with the chunk delimiter and then running the M2M algorithm on those sequence strings to learn alignments allows for the delimiter chunk to be included in chunks with the actual character data. This approach does not guarantee that DirecTL+ will predict exactly one lemma for each stem.

to the WORDLEN, STARTOFSEQUENCE, NEARBEG, ENDOFSEQUENCE, or NEAREND feature templates.

n-gram	Examples
1-grams, word	{ *'.z.q*, *w*, *p* }
2-grams, word	{ (*'.z.q*, *w*), (*w*, *p*) }
1-grams, local	{ . . . , *m*, *l.k*, *#*, *'.z.q*, *w*, *p*, . . . }
2-grams, local	{ . . . , (*m*, *l.k*), (*l.k*, *#*), (*#*, *'.z.q*), (*'.z.q*, *w*), (*w*, *p*), . . . }

transition	Examples
order-1, both	*z.q*

linear chain	Examples
1-grams, word	{ [*'.z.q*, *z.q*], [*w*, *z.q*], [*p*, *z.q*] }
2-grams, word	{ [(*'.z.q*, *w*), *z.q*], [(*w*, *p*), *z.q*] }
1-grams, local	{ . . . [*m*, *z.q*], [*l.k*, *z.q*], [*#*, *z.q*], [*'.z.q*, *z.q*], [*w*, *z.q*], [*p*, *z.q*] }
2-grams, local	{ . . . [(*m*, *l.k*), *z.q*], [(*l.k*, *#*), *z.q*], [(*#*, *'.z.q*), *z.q*], [(*'.z.q*, *w*), *z.q*], [(*w*, *p*), *z.q*] . . . }

joint n-gram	Examples
1-grams, word	[*'.z.q*, *z.q*]
2-grams, word	none
1-grams, local	{ . . . , [*m*, *m*], [*l.k*, *l.k.'*], [*#*, *#*], [*'.z.q*, *z.q*] }
2-grams, local	{ . . . , ([*m*, *m*], [*l.k*, *l.k.'*]), ([*l.k*, *l.k.'*], [*#*, *#*]), ([*#*, *#*], [*'.z.q*, *z.q*]) }

Table 6.4: DirecTL+ feature template examples. Examples given are relative to the second input chunk, *w*, in the M2M pre-processed undiacritized Syriac stem *'.z.q-w-p* in its context of John 19:15. Bigrams are indicated by (*chunk-1*, *chunk-2*); unigrams have no surrounding parentheses; n-grams where $n > 2$ are not shown. Pairings of input and output chunks are indicated by [*input-chunk*, *output-chunk*]

DirecTL+ Parameters

To determine reasonable parameters for the M2M alignment tool we conducted a grid search. We searched over the cross-product of the maxX [1,5], maxY [1,5], delX yes/no, and delY yes/no parameters. For each parameter profile, M2M is run on the development training data, then the aligned data is used to train a DirecTL+/W instance which has only a minimal set of n-gram features (context size of 3). Finally, the prediction accuracy is measured on a held-out validation set. Using 75% of the PNT/U development data for training and 25% for testing, the best M2M parameters are maxX = 3, maxY = 3, delX = yes, delY = no. We conducted the grid search only on the PNT/U development data and for word context stem-to-lemma linkage, but the resultant M2M parameters seem to work well for the other data sets and for local context stem-to-lemma linkage.

To determine reasonable parameters for DirecTL+, given the chosen M2M parameters, we again

conducted a grid search. We searched over the cross-product of the n-gram context window size c [1,5], transition features on/off (on: Markov order 1; off: Markov order 0), linear chain features on/off (uses the same c as the n-gram template), and joint n-gram maximum n-gram size n_{joint} [0,5]. The maximum n-gram size $n_{standard}$ for the n-gram feature template is always set to $2c+1$ to extract every possible n-gram from the context: for $c = 1$, $n_{standard} = 3$ and for $c = 5$, $n_{standard} = 11$. We selectively searched among the parameter profiles by adding the feature types in the order n-gram, transition, linear chain, and joint n-gram and then tried modulations on well-performing parameter profiles to exhaustively search out its neighborhood. Using 75% of the PNT/U development data for training and 25% for testing, a good set of parameters for DirecTL+/W is $c = 5$, hence $n_{standard} = 11$ for standard n-gram features, transition features = on, linear chain features = on, and $n_{joint} = 2$ for joint n-gram features. DirecTL+ always employs a beam search when joint n-gram features are used; we kept the default beam size of 20. Similarly, a good set of parameters for DirecTL+/L is $c = 4$, hence $n_{standard} = 9$, transition features = on, and linear chain features = on. Using joint n-gram features ($n_{joint} > 0$) did not improve accuracy and drastically increased training and prediction time.

6.4 Results

In this section we report the performance of the models described in the previous section on the stem-to-lemma linkage task for each of the four data sets. The gold-standard stems are used as test input, as opposed to running SyroMorph's segmentation module and taking its predicted stems as test input. To conduct better error analysis on unknown stems we use an 80-20 development-blind split of the data (see Table 5.3), whereas previous evaluations of stem-to-lemma linkage on the PNT have used a 90-10 split (McClanahan et al., 2010; Lindgren, 2011). Using 80% of the PNT data for development tests instead of 90% should not hurt model performance too much inasmuch as training on 35% of the data yields high performance on a variety of morphological tasks on the PNT (McClanahan et al., 2010). We use a five-fold cross-validation experimental regime, first on the development set and then on the blind test set. Reported accuracy and training time values are averages over the five folds, and error bars indicate minimum and maximum values. In order to better analyze the characteristics of each model, we break the accuracy results down by overall, unknown, known unambiguous, and known ambiguous accuracy similar to the corpus ambiguity analysis presented in Table 6.1. Table 6.5 consolidates symbols and terminology for a convenient reference and as a

common key to the figures in this section.

Symbol	Meaning
QC	Data is the Quranic Corpus (Arabic)
PNT	Data is the Peshitta New Testament (Syriac)
/U	Data includes no diacritics
/D	Data includes all available diacritics
Baseline	Stem-as-lemma heuristic for unknown stems; memorizer for known stems
Morfette	Edit script tags in an MEMM for all stems
Hybrid-Morfette	Morfette for unknown stems; memorizer for known stems; maxent model for known ambiguous stems
DirecTL+	PHMM and monotone phrasal decoder for string transduction for all stems
/W	Model uses only word-context features
/L	Model uses local-context and word-context features
Error bars	Min and max values of five-fold cross-validation

Table 6.5: Symbol Reference. Symbols and terminology for interpreting Figures 6.3–6.8

6.4.1 Development Set Experiments

We first analyze model performance on the development set. In the five-fold cross-validation the test set is rotated such that each fold of the development data is used for testing. The "available" training data is the combination of the four folds not being used as the test set. We report accuracies for models trained on progressive amounts of data, visualized as learning curves. We similarly report model training time in a learning curve. Our analysis focuses on model performance with small amounts of training data, between 0.1% and 20% of the available training data. We also evaluate at 100% of the available training data in order to indicate how well the models scale given more training data.

Overall, the hybrid-Morfette model is the best performer. Figure 6.3 shows the prediction accuracy of the models on all stems in the development test data. Figure 6.3a shows the model performance when using word-context features, and Figure 6.3b shows the model performance when using local-context and word-context features. In most cases, using local-context features does not significantly improve (or hinder) the models' predictive accuracy. In all cases, the hybrid-Morfette model surpasses the baseline accuracy within the first 1% of the available training data and achieves accuracy levels that are as high as—or higher than—the other models at all points on the learning curve, although the differences towards the end of the curve tend to be small. The performance of the Morfette model is nearly identical to that of the hybrid-Morfette model, although it tends to be very slightly lower after about 10% of the data is seen.

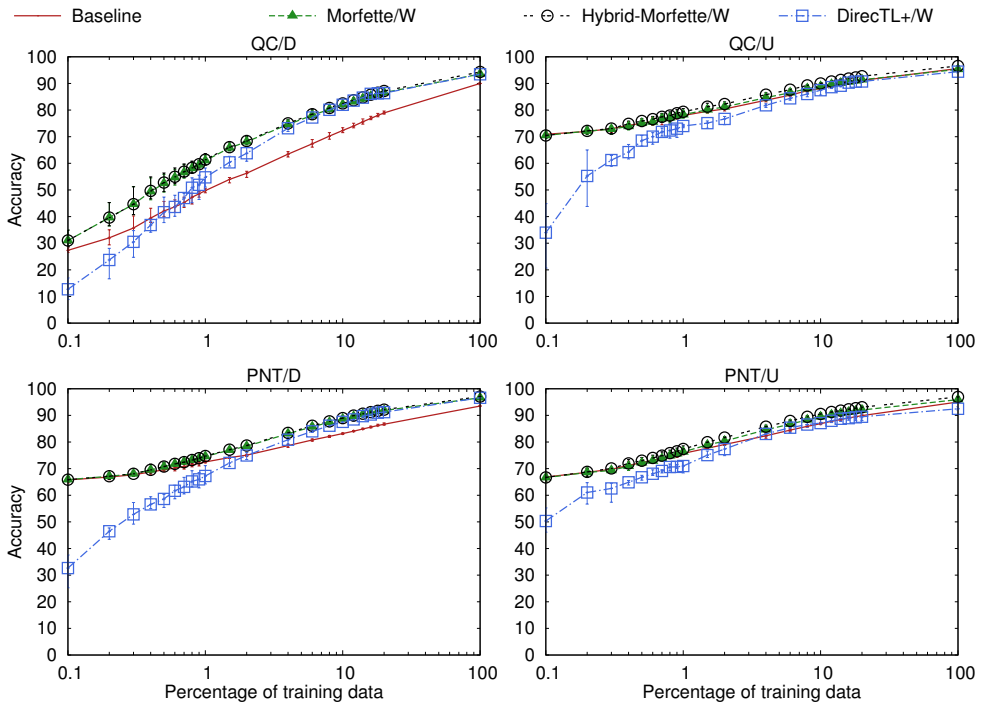
The DirecTL+ model provides sub-baseline predictions for the early, low-resource portion of the annotation project. The DirecTL+ model does not approach the trajectory of the Morfette and hybrid-Morfette models until at least around 6-10% of the training data is used. One situation where local-context features do improve performance is for DirecTL+ on the PNT/U data set. With only word-context features, DirecTL+ does not perform significantly better than the baseline, and in fact, with all of the training data, does worse than the baseline. With local-context features, DirecTL+ converges to a similar level of performance as the rest of the algorithms at the high-resource end of the learning curve.

All of the models, as they are currently implemented, can provide lemma predictions for a single verse at interactive speeds, but the time to train the models from scratch is impractical for interactive model refinement with larger and larger amounts of training data. Prediction times (not plotted here) are suitable for interactivity at well under one second per verse for all models. Figure 6.4 plots the time required to train each model on various amounts of data when using local-context features; the training times when using only word-context features have similar trends (not shown), although out at 100% of the training data the training times are about an order of magnitude smaller than for the local-context model counterparts (note the logarithmically-scaled y-axis). The baseline model's training time appears to be nearly constant; the training times for the other models increase exponentially which appears linear in the log-log plot. The training times for Morfette and hybrid-Morfette are a full order of magnitude faster than those of DirecTL+. Unfortunately, starting somewhere between 1% and 10% of the available training data, none of the models except for the baseline can currently be trained from scratch quickly enough to allow for interactive model refinement at our desired 10 second parallel training mark. It is possible that model training time would be reduced if the models were instead trained incrementally, initializing from a previous instance of the model trained with less data. We address incremental training in the next chapter.

To further understand the nature and performance of the models we analyze how the models do on the unknown, known unambiguous, and known ambiguous subsets of the test data. The ambiguity analysis presented in Figure 6.1 showed that at the low-resource end of the learning curve the unknown subset accounts for most of the test set until about 0.7% to 2%, after which the known unambiguous subset accounts for most of the test set.

For the unknown stems, the Morfette and hybrid-Morfette models are consistently the best performers. Figure 6.5 plots prediction accuracy on the unknown subset of the test data for the models trained with

(a) Word-context features



(b) Local-context and word-context features

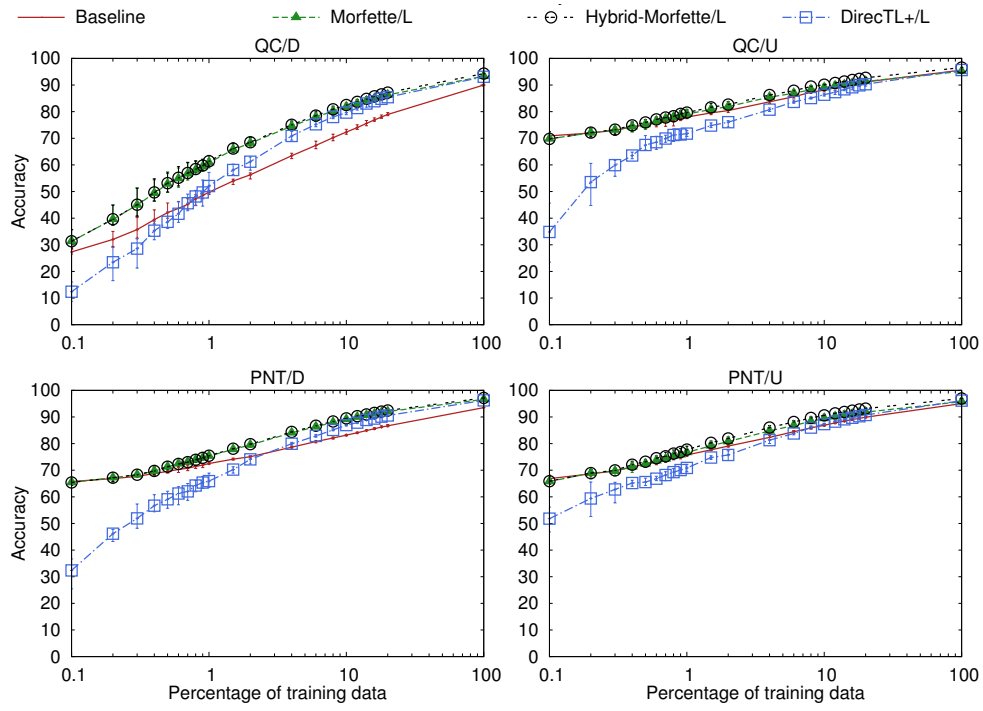


Figure 6.3: Overall Accuracy. Average model prediction accuracy on all stems in the development test data

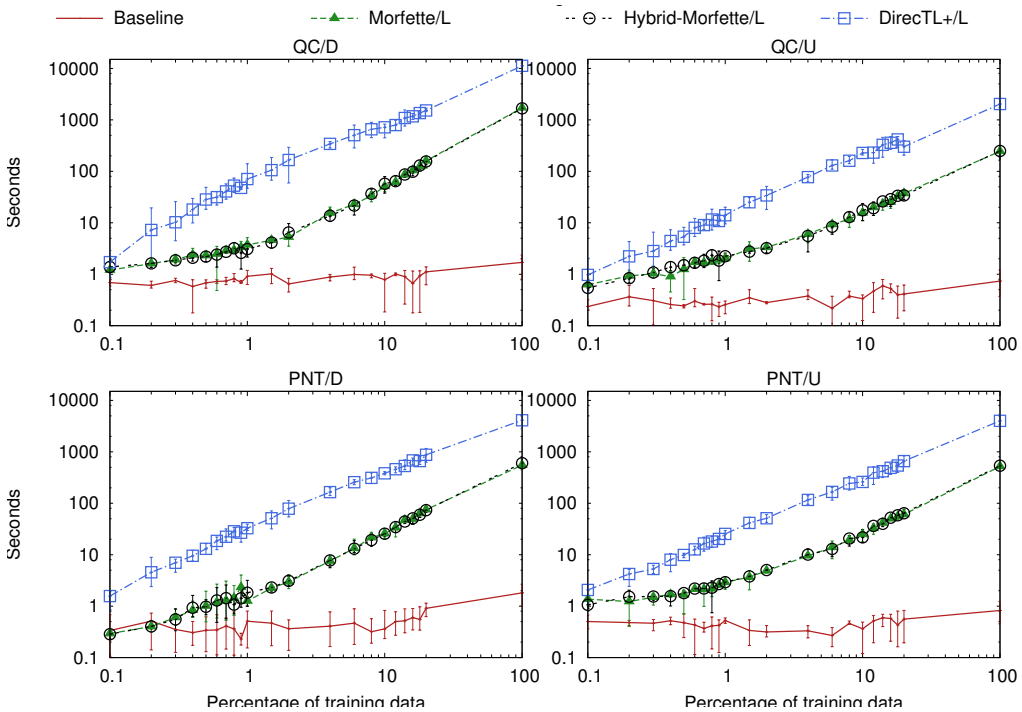


Figure 6.4: Training Time. Average local-context model training time on the training data. Note that the y-axis is logarithmically scaled

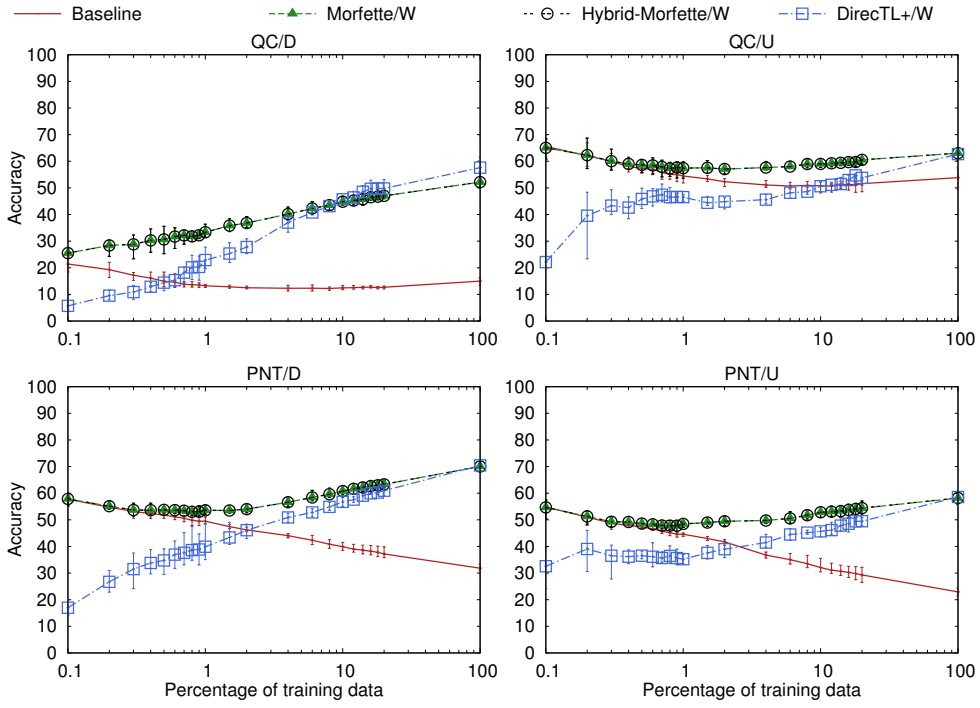
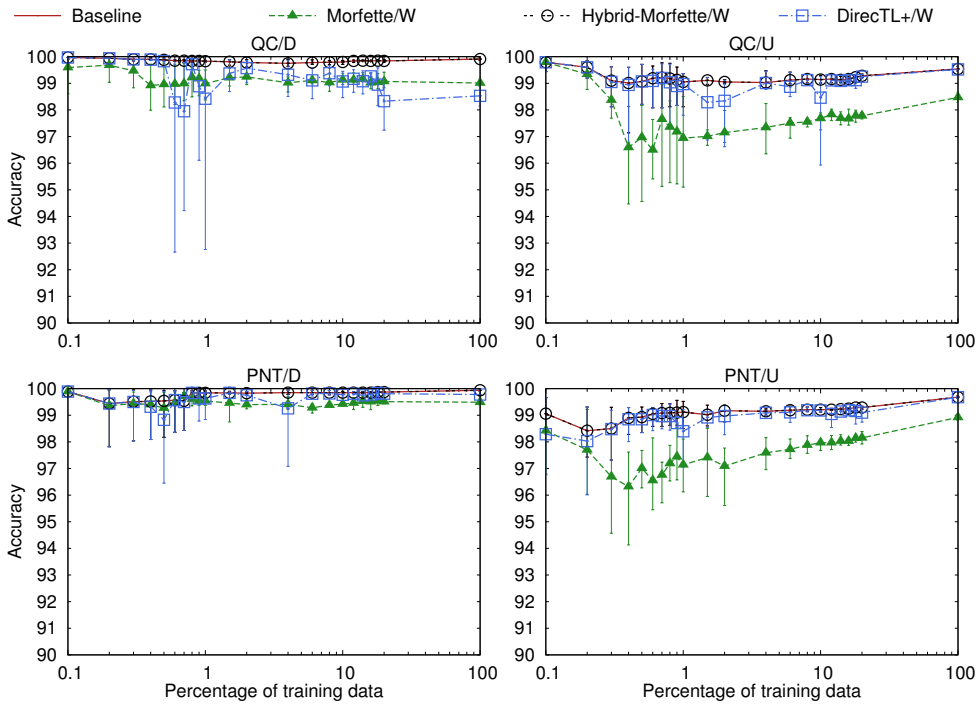


Figure 6.5: Unknown accuracy. Average word-context model prediction accuracy on the development test data for stems not seen in the training data

(a) Word-context features



(b) Local-context and word-context features

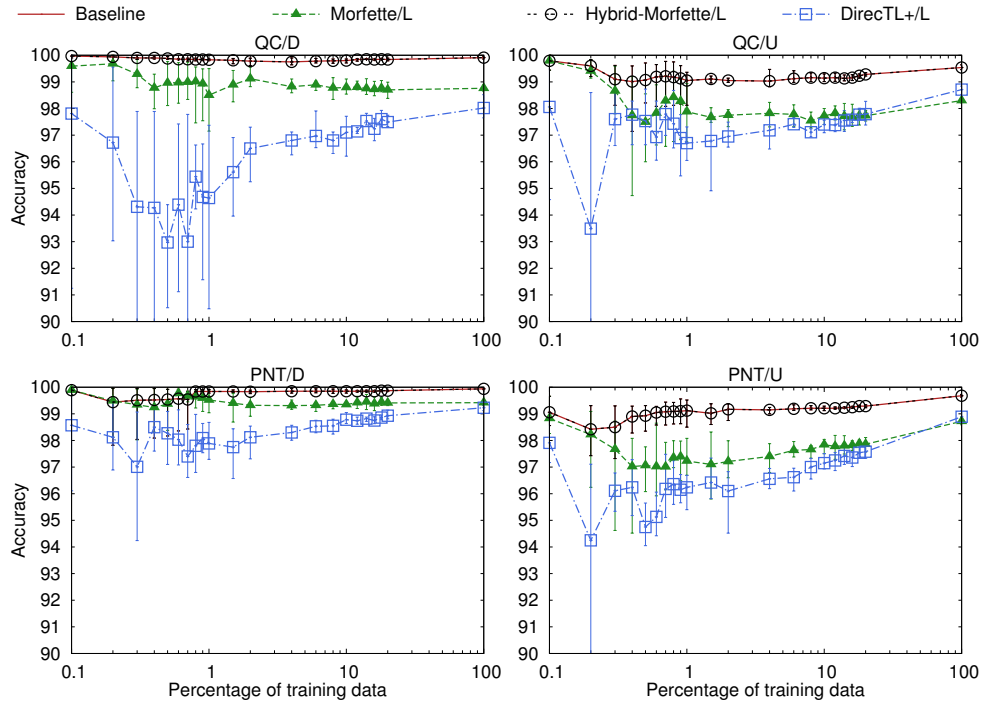
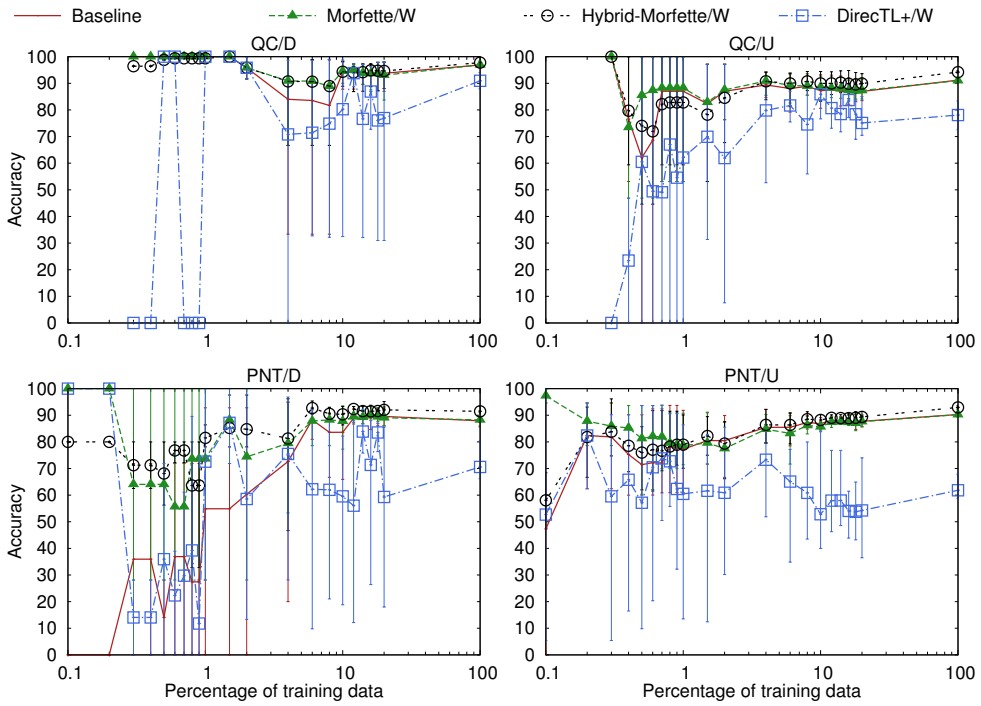


Figure 6.6: Known Unambiguous Accuracy. Average model prediction accuracy on the development test data for stems seen with only a single lemma type in the training data. Note that the y-scale on the plots is from 90–100%

(a) Word-context features



(b) Local-context and word-context features

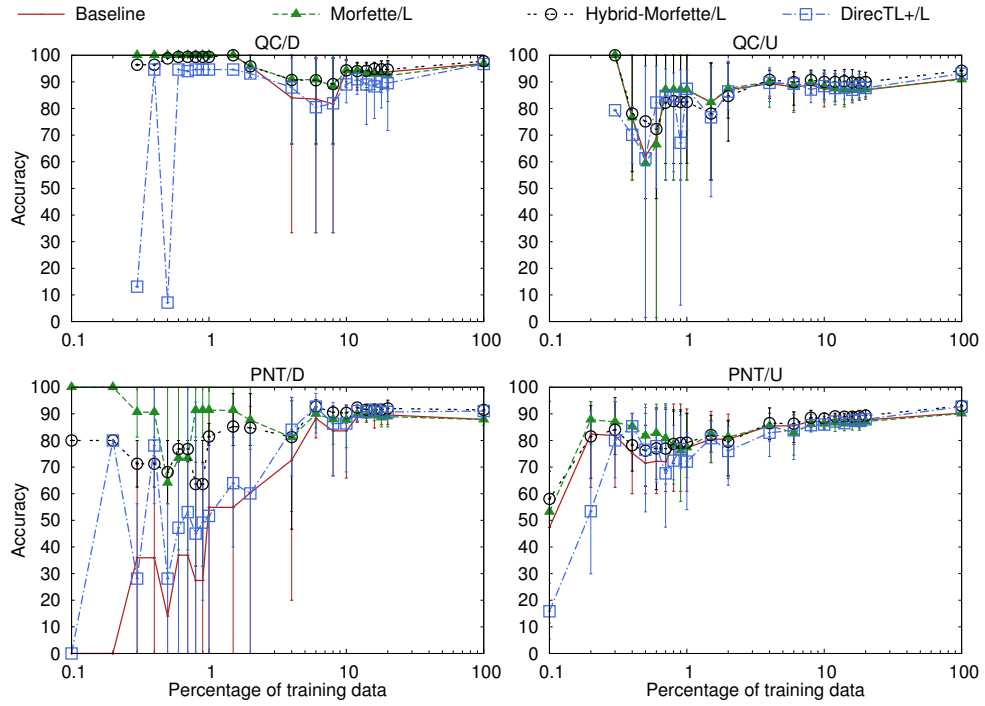


Figure 6.7: Known Ambiguous Accuracy. Average model prediction accuracy on the development test data for stems seen with multiple lemma types in the training data. Note that the x-scale starts at 1% instead of 0.1%

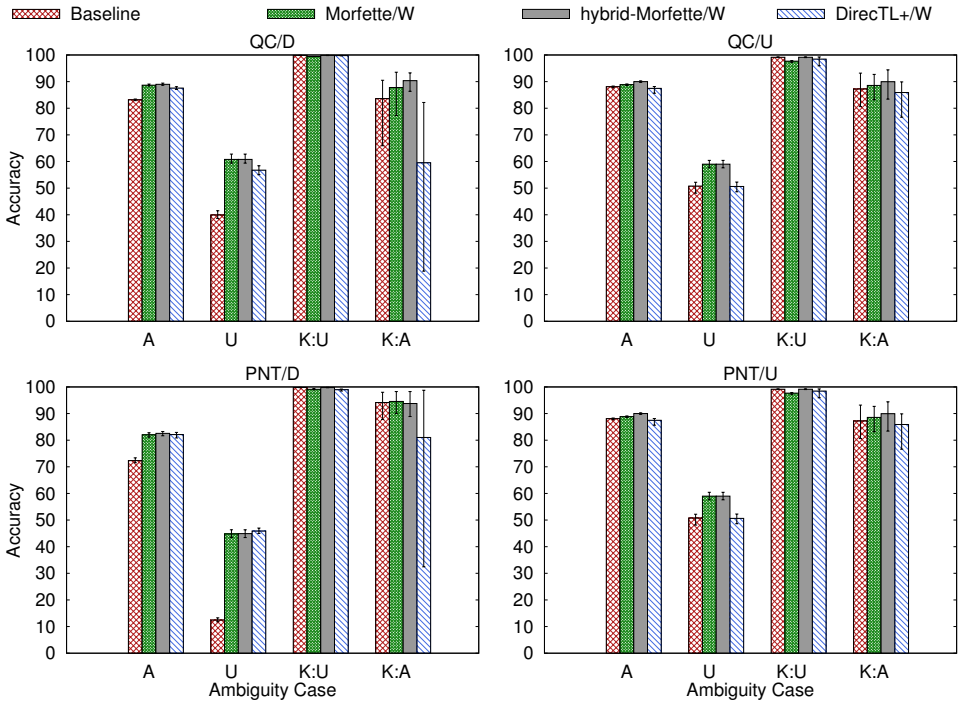


Figure 6.8: Accuracy on Blind Test Set. Average model prediction accuracy on the blind test data for ambiguity cases: all (A), unknown (U), known unambiguous (K:U), and known ambiguous (K:A)

only word-context features; the performance of the models trained with local-context features is not very different visually so the plot is omitted. A model's prediction accuracy on unknown stems indicates how well it is able to generalize its knowledge of morphological patterns when memorization is not possible. The sophisticated models are designed to generalize morphological patterns and so should perform much better than the baseline. Indeed, the Morfette model (which hybrid-Morfette uses as its component model to handle unknown stems), performs the same as, or better than, the baseline over the whole learning curve. The closeness in accuracy between Morfette and the baseline for small amounts of training data indicates that the most frequent edit script is equivalent to the baseline's heuristic of predicting the stem as the lemma, and that Morfette learns to predict that edit script. Given more training data, however, Morfette is able to associate edit scripts other than the stem-as-lemma script with certain stem patterns and thus it performs significantly better than the baseline starting between 1% to 6% over all of the data sets. DirecTL+ on the other hand performs worse than the baseline early on and never surpasses Morfette except on the QC/D data set. This disparity between DirecTL+ and the baseline indicates that DirecTL+ is not simply learning character

sub-sequence pass-through transformations. While we noted that DirecTL+ had a potential advantage over Morfette by learning disjoint insert and delete operations instead of packaged operations in an edit script, that potential was not realized in most of the experiments.

For known unambiguous stems, the baseline and hybrid-Morfette do the best using memorization. Figure 6.6 plots model prediction accuracy on the known unambiguous subset of the test data, broken down by models using word-context features (Figure 6.6a) and those using local-context and word-context features (Figure 6.6b). The baseline model and the hybrid-Morfette models are precisely equivalent in this setting since they use the same memorizer mechanism to predict the single known lemma for the stem. The extremely high accuracy that memorization attains indicates that stems which are unambiguous in the training data infrequently have a new lemma in the test data. In principle, with the right features, more sophisticated models than the memorizer could predict when a known unambiguous stem should become ambiguous and propose lemmas for the stem. However, predicting the emergence of ambiguity seems impractical given the small number of such stems which become more ambiguous and given the sub-baseline performance of DirecTL+ and Morfette. DirecTL+ with word-context features almost matches the baseline accuracy at many points in all of the data sets except QC/D, indicating that DirecTL+/W has a tendency to memorize the data. However, this tendency to memorize is not as strong when local-context features are used. We hypothesize that DirecTL+/L does not memorize the data as much as using only word-context features because the greater diversity in the local-context features, and hence small occurrence frequency, prevents the PHMM training procedure from saturating those weights as much as for the smaller set of word-context features, which occur more frequently. Unlike for other machine learning problems with more inherent ambiguity in the problem, memorization is an effective method for a good portion of the stem-to-lemma linkage problem.

For known ambiguous stems, the Morfette and hybrid-Morfette models consistently perform on par with the strong-performing baseline. Figure 6.7 plots model prediction accuracy on the known ambiguous subset of the test data, broken down by models using word-context features (Figure 6.7a) and those using both local-context and word-context features (Figure 6.7b). Because of the small number of known ambiguous stems (see Figure 6.1) substantial noise results. In fact, the learning curves start at 1% instead of 0.1% to ensure that there are at least a few instances of known ambiguous stems. The baseline approach of memorizing the most frequent lemma that occurs with a given stem in the training data results in an average baseline accuracy above 80% over all of the learning curves except for PNT/D; the accuracy stays around 90% after 10% of the training data in all data sets. The fact that the baseline model performs so well

indicates that even nominally ambiguous stems are usually dominated by one lemma. The Morfette and hybrid-Morfette models consistently achieve the same or higher average accuracy as the baseline, but given the min and max bounds and the small number of repetitions (five) the difference is not statistically significant. Remember that the difference between Hybrid-Morfette/W and Hybrid-Morfette/L is in the feature set used in the Morfette component to handle unknown stems, not in the specialized maxent models for each known ambiguous stem type, which always uses local-context features. However, hybrid-Morfette does not gain an advantage over Morfette with the specialized models. The DirecTL+ model has a strong tendency to not memorize when using only word-context features as seen by the sub-baseline average accuracy, but a strong tendency to memorize when using local-context and word-context features as seen by an average accuracy close to the baseline. This is the reverse of what was observed for known unambiguous stems in Figure 6.6. We hypothesize that the difference when using only word-context features is that the ambiguity for ambiguous stems cannot be perfectly memorized because the feature weights are kept about equal for the competing alignments. When using local-context features we hypothesize that there is a set of rare features which occur almost exclusively with the ambiguous stems and are therefore highly indicative of which character sub-sequence alignment to use.

6.4.2 Blind Set Experiments

To further validate model performance and to ensure that we have not over-tuned our models to the development set, we evaluate on a blind test set which was held in reserve during the model development process. In the cross-validation development data, all models seem to reach a steady level of performance by the time they are trained on 10% of the available training data which is about 399 annotated verses in the QC/* data sets and 509 verses in the PNT/* data sets. The question we now consider is how well the models might do at automatically annotating the "rest" of the unannotated data if the human annotators stopped manual revision of the annotated corpus at this point, where the "rest" of the unannotated data is the blind test set. We use the same five training folds from the development data to train the models with 10% of the available data, and then evaluate on the blind test set so we are still able to give min and max bounds on the prediction accuracies. In terms of the number of verses, the blind test set is 1.25 times larger than any of the test folds used in the development cross-validation.

The model performance on the blind test set is consistent with the results on the development set, and hybrid-Morfette is consistently a top performer. Figure 6.8 shows the predictive accuracy of the models on

the blind test set when using only word-context features; the results of using local-context and word-context features are similar so are not shown. The level of accuracy agrees with the results in cross-validation at the 10% training data mark. For all data sets, hybrid-Morfette achieves the highest average overall accuracy, and with statistical significance on the undiacritized data sets since its min bound is above the other model's max bounds. hybrid-Morfette and Morfette do best in unknown accuracy on all of the data sets, always achieving well above the baseline accuracy and usually significantly better than DirecTL+ as well, except for on the PNT/D. hybrid-Morfette uses the same memorization mechanism as the baseline for known unambiguous stems which achieves the highest accuracy. hybrid-Morfette achieves the highest average accuracy for known ambiguous stems but the min and max bounds overlap with all of the other models.

6.5 Conclusion

In this chapter we evaluated several models for stem-to-lemma linkage on the Quranic Arabic Corpus and Syriac Peshitta New Testament with the goal of identifying the methods most suitable for corpus-dictionary linkage and especially for interactive pre-annotation assistance. We evaluated the baseline, Morfette, hybrid-Morfette, and DirecTL+ models by sweeping out five-fold cross-validated learning curves on development data. We also evaluated those methods on the blind test data using 10% of the training data from the development training folds. The baseline model, which uses a heuristic for unknown stems and a memorizer for known stems, achieves high accuracy given sufficient training data. For example, it usually achieves above 80% accuracy using 10% of the training data. The baseline model can be trained and retrained quickly since it simply requires counting frequencies of stem-lemma pairs in the test data. This fact makes the baseline model a good initial candidate for full interactive pre-annotation assistance. However, more complex machine-learning models outperform the baseline, particularly when little data is available and when dealing with unknown stems. For small amounts of annotated training data Morfette and hybrid-Morfette consistently do the best, but for larger amounts of training data with more known unambiguous stems hybrid-Morfette does slightly better by simply memorizing the single lemma for the stems. DirecTL+ requires between 1% and 10% of the training data to begin to compete with the other models, but it never significantly surpasses the other models in terms of overall accuracy. Another point in favor of hybrid-Morfette is that our current implementation can be trained on larger amounts of training data more quickly than can DirecTL+. However, when large amounts of training data are available training hybrid-Morfette from

scratch takes a few seconds to minutes which is not suitable to support interactive pre-annotation refinement via iterative model retraining.

Chapter 7

Interactive Pre-annotation Assistance for Corpus-Dictionary Linkage

In this chapter we develop context-sensitive pre-annotation assistance for corpus-dictionary linkage (CDL) which achieves high prediction quality and which can be trained sufficiently quickly to be interactive. The key contribution to facilitate interactive pre-annotation assistance is to use an incremental algorithm to train the model for CDL which avoids wasting previous computation like training from scratch would do. The algorithm to simulate CDL annotation and the experiment scenarios of interest are discussed in Section 7.1. The baseline CDL pipeline and hybrid maximum entropy Markov model (MEMM) pipeline, which incorporates the hybrid-Morfette stem-to-lemma linker from Chapter 6, are presented in Section 7.2. An incremental training algorithm which is able to learn from partially annotated sequences is developed for the hybrid MEMM pipeline in Section 7.3. Further experiment details are given in Section 7.4. The experiment results are analyzed in Section 7.5. Ultimately, in Section 7.6, we conclude that incrementally training the hybrid pipeline is interactively viable and usually requires 2,000 or fewer annotated tokens before achieving and maintaining better accuracy than the baseline pipeline.

7.1 Simulation Experiment Scenarios

To evaluate the interactive viability of the different CDL pre-annotation methods we simulate annotating each word token in a corpus with a link to the dictionary. The simulation procedure is sketched in Algorithm 1. We will discuss the simulation parameters in greater detail after reviewing the structure and flow of the simulation. The order in which the simulated annotator traverses and annotates the corpus is controlled by a data selection strategy, such as annotating tokens in reading order (line 3). The data selection strategy iterates over decision points which are a combination of token ID and the type of annotation to be provided: morphological segmentation, lemma, or dictionary entry (line 4). The model update strategy controls when the model is updated, such as when the sequence changes (line 5). The model is updated with the accumula-

tion of decisions since the last model update (lines 2, 6, and 15). Periodically the model's prediction quality is measured on a held-out data set (line 8). The model is queried to pre-annotate the decision point's whole containing sequence (line 11). The simulated annotator is then queried to accept or correct the annotation for the current decision point given the pre-annotations for the whole sequence (line 12). Given the annotator's decision, the progressive validation (i.e., online) prediction quality measures are updated (line 13) and the annotation is recorded in the corpus (line 14). Finally, after the whole corpus has been annotated, the model is updated for a last time so that held-out measures can be calculated using the most up-to-date model (lines 18 and 19).

Algorithm 1 Simulate interactive, machine-assisted annotation project

Input: current working *corpus*; simulated *annotator*; CDL pre-annotation *model*; data selection strategy *dataSelector*; model update strategy *updateSchedule*
Output: *corpus* has been annotated

```
1:  lastDecPoint ← null
2:  decisionsSinceLastUpdate ← ∅
3:  while dataSelector.hasAnotherPointIn(corpus) do
4:      decPoint ← dataSelector.getNextPointIn(corpus)
5:      if updateSchedule.shouldUpdate(lastDecPoint, decPoint) then
6:          model.update(decisionsSinceLastUpdate)
7:          record training time
8:          periodically compute and record held-out quality measures
9:          decisionsSinceLastUpdate ← ∅
10:     end if
11:     preAnnotation ← model.preAnnotate(corpus.getSequence(decPoint.token))
12:     decision ← annotator.getDecision(decPoint, preAnnotation)
13:     record progressive validation quality measures given decision.annotation and preAnnotation
14:     corpus.addAnnotationFor(decPoint, decision.annotation)
15:     decisionsSinceLastUpdate ← decisionsSinceLastUpdate ∪ {decision}
16:     lastDecPoint ← decPoint
17: end while
18: model.update(decisionsSinceLastUpdate)
19: record training time and held-out quality measures
```

We explore three data selection strategies. The first strategy selects items in **reading order** (RO) by traversing sequences either at random or in the natural order, and traversing the tokens within sequences strictly sequentially. Annotating while reading through the corpus helps to maintain topical context in the annotator's mind and is an easy way to manually track progress in the annotation effort. The **descending word type frequency** (DWF) strategy indexes the frequency of word types and then iterates through the word tokens in descending word type frequency order. The DWF strategy mimics an annotator focusing

first on the most frequent word types, which annotators may wish to do in practice so as to maintain focus on one word type at a time and thus allow them to quickly annotate many tokens throughout the corpus. Combined with interactive pre-annotation assistance, this data selection strategy should rapidly result in uniformly annotated function words and allow the annotator to study and/or annotate other frequent content words. The **ascending word type frequency** (AWF) strategy is like the descending word type frequency strategy except it iterates through the word tokens in ascending word type frequency order. The AWF strategy is not compelling for real-world application, apart from focusing on the rare or obscure parts of the corpus, but it does serve to evaluate the generalization abilities of a model when trained early on with many singleton instances of diverse input types. We assume that the descending or ascending word frequency strategies are used with a keyword-in-context (KWIC) view which presents all of the instances of the word type throughout the corpus along with their context; for example, see Figure 1.1b. Other data selection strategies are possible, including active learning strategies that use the model to select the next unannotated item for the humans to annotate (e.g., Haertel et al., 2010a), but we only explore the RO, DWF, and AWF data selection strategies in this work.

We employ three model update strategies. Model update strategies are implemented by looking for changes between the last decision point and the current decision point. The first strategy updates the model whenever the sequence has changed (SEQ). When combined with the reading order data selection strategy, the sequence change update strategy does not interactively improve the model, yet this scenario does reduce the number of total model updates and the training algorithm is ever only applied to fully annotated sequences. The second model update strategy updates the model whenever the token changes (TOK). The token change strategy has more interactive potential than the sequence change strategy, provided the time to update the model is short. Combined with any of the data selection strategies, using the token change update strategy requires that the training algorithm be able to learn from partially annotated sequences. The third model update strategy updates the model whenever the word type changes (WTC). When combined with the descending or ascending word type frequency data selection strategies, the word type change update strategy only updates the model after all of the tokens of that word type have been annotated. In effect, the word frequency and word type change scenarios simulates the annotator submitting annotations for the whole token set at one time, otherwise the model would have had a chance to update sooner. Other model update strategies are possible, such as strategies that take into account average model training time, idle time, and so on, but we only explore the SEQ, TOK, and WTC model update strategies in this work.

We experiment with six scenarios, defined by the combination of data selection strategy and model update strategy, in three evaluation configurations. The grid in the left part of Table 7.1 shows the cross product of the three data selection strategies and three model update strategies, where numbers indicate which evaluation configuration the scenario is applicable to and "NA" indicates that the scenario is not applicable since it is redundant with another one. The applicable experiment scenarios are: RO+SEQ, RO+TOK, DWF+TOK, DWF+WTC, AWF+TOK, and AWF+WTC. The right part of Table 7.1 summarizes the evaluation configurations by simulation mode, model training algorithm, and the kind of annotated training sequence.[1] In the **snapshot** simulation mode the model is trained on training data subsets of increasing size and is evaluated on a fixed held-out set, thus providing a "snapshot" of model performance. In the **complete** simulation mode the annotation of each token is simulated, the model's accuracy is measured after every simulated decision, and the model is trained according to the model update strategy. In configuration 1 we do snapshot simulation and train the CDL pipeline from scratch on each training data subset which contain fully annotated sequences. By its nature, configuration 1 provides a snapshot of the hypothetical RO+SEQ experiment scenario where sequences are fully annotated in reading order and the model is retrained from scratch, instead of incrementally, when the sequence changes. In configuration 2 we simulate complete annotation of every decision point in the RO+SEQ scenario and we employ incremental training for fully annotated sequences. In configuration 3 we simulate complete annotation of every decision point in the other five experiment scenarios and we employ incremental training for partially annotated sequences. Further experiment details, including applicable accuracy and training time measurements for each scenario, are discussed in Section 7.4.

SCENARIOS

Data Selection	**Model Update** SEQ	TOK	WTC
RO	1,2	3	NA
DWF	NA	3	3
AWF	NA	3	3

CONFIGURATIONS

ID	**Mode**	**Algorithm**	**Sequence**
1	snapshot	from scratch	full
2	complete	incremental	full
3	complete	incremental	partial

Table 7.1: Experiment scenarios divided into three evaluation configurations. **SCENARIOS**. An experiment scenario is a combination of a data selection strategy and a model update strategy. Each experiment scenario is evaluated in the configurations indicated by the configuration IDs in the scenario cell. Cells marked "NA" are redundant with other experiment scenarios and are therefore not evaluated. **CONFIGURATIONS**. Each configuration specifies the simulation mode, the training algorithm type, and the annotated sequence type used to evaluate an experiment scenario.

[1]The order of the configurations reflects a principled adaptation of the batch-style SyroMorph machine learning codebase to support online, incremental learning such that complete simulation of every decision point can be done.

7.2 CDL Pipeline Models

In this section we discuss the pipeline models for CDL which we use to provide pre-annotations in the annotation simulation experiments.

The CDL formulation we employ for Classical Syriac and Quranic Arabic is a pipeline of three subtasks: morphological segmentation (MS), stem-to-lemma linkage (STLL), and homographic headword disambiguation (HHD). Figure 7.1 shows how the pipeline is structured, with the MS subtask feeding into the STLL subtask, and the STLL subtask into the HHD subtask. We adapted and extended the SyroMorph code base (McClanahan et al., 2010) to perform CDL following this pipeline structure. The inputs to the pipeline are word tokens and the outputs are segmentation, lemma, and dictionary key information for each token. The inputs to the MS subtask are word tokens and the outputs are segmentation patterns which identify the prefix, stem, and suffix of the corresponding input token. Multiple prefixes in a token are concatenated into a single prefix, and similarly for multiple stems, and suffixes. The inputs to the STLL subtask are stems and the outputs are lemmas (as explored in Chapter 6). Tokens with multiple stems also concatenate the multiple lemmas into a single lemma. The inputs to the HHD subtask are lemmas and the outputs are dictionary keys which are a combination of the lemma (headword) and a numeric identifier to distinguish entries with the same headword from one another. In the experiments, the numeric ID is backed by part of speech tags since that is the content the simulated annotator puts in dictionary entries; however, the HHD model is not aware of the content or structure of the dictionary entries, only of the fact that there are multiple distinct entries with the same headword.

We experiment with two CDL pipelines: a non-context-sensitive baseline and a context-sensitive hybrid maximum entropy Markov model (MEMM) pipeline. The baseline pipeline employs heuristics and most frequent empirical statistics from training data. The hybrid MEMM pipeline extends the baseline with maximum entropy (MaxEnt) scoring models to achieve context sensitivity. The structure and features of each pipeline type are discussed in turn in the following sub-sections.

7.2.1 Baseline Pipeline

The baseline pipeline is a set of classifiers, one for each CDL subtask. More precisely, each model in the pipeline is a degenerate sequence tagger which simply uses a classifier for each token in the sequence. Each subtask's classifier is a hybrid model with two components distinguished by their use at prediction time:

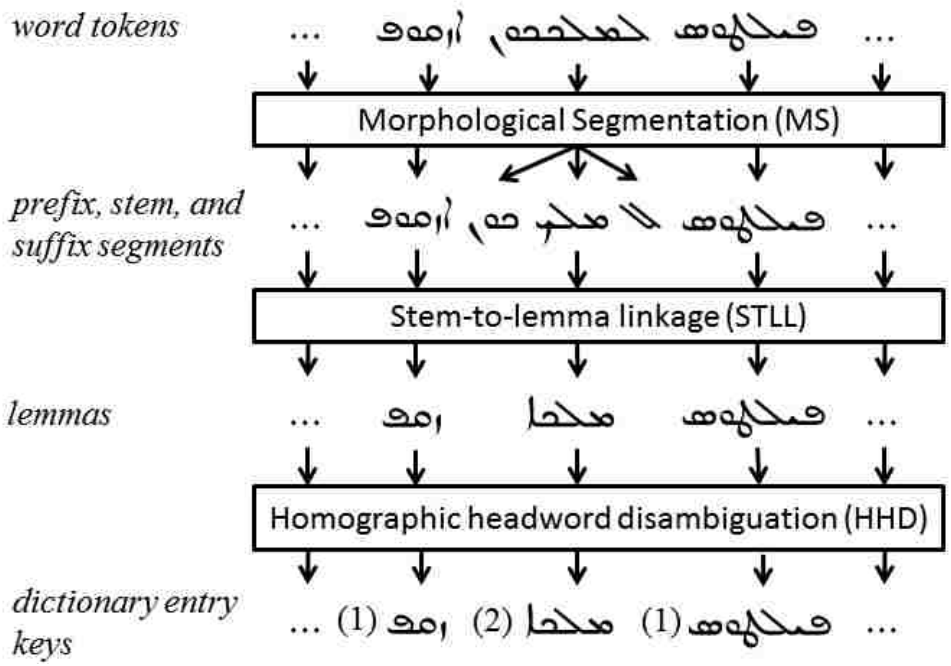


Figure 7.1: An illustration of pipeline structure used for corpus-dictionary linkage. Example instances of each input/output type are given from John 19:15 of the Syriac PNT. The numeric identifiers in the dictionary entry keys are for demonstration purpose and do not necessarily reflect identifiers used in a particular simulation.

one component handles labeling known input types, and the other component handles labeling unknown input types. Each subtask's known component model is a "memorizer". The memorizer is trained simply by counting the occurrences of each input-label type pair in the labeled training data. For a known input type the memorizer predicts the most frequent label associated with that input type. Each subtask's unknown component model is a heuristic rule. The unknown component for the MS subtask greedily segments an input token by using the longest prefix and suffix character sub-sequences of the token which are known to be valid morphological affixes (given the labeled training data) such that the remaining stem of the token is not empty. The unknown component for the STLL subtask simply predicts the input stem as the output lemma. The unknown component for the HHD subtask predicts that the lemma should be added to the dictionary as the first entry with that headword.

7.2.2 Hybrid MEMM Pipeline

The hybrid MEMM pipeline is a set of sequence taggers, one for each CDL subtask. At prediction time, tag sequences are decoded in reading order thus the MaxEnt-based local scoring models can use Markov features, that is, previous tags. Each CDL subtask's local scoring model is a hybrid MaxEnt model with three components distinguished by their use at prediction time: one component handles labeling known ambiguous input types, another component handles labeling known unambiguous input types, and the last component handles labeling unknown input types. For example, the hybrid-Morfette model for STLL is an instance of such a hybrid (see Section 6.3.3). Each subtask's known ambiguous component model is a set of MaxEnt models, one model for each input type known to have two or more labels and which is trained on all of the labeled instances of the input type; the feature templates are discussed later. Each subtask's known unambiguous component model is a memorizer as in the baseline pipeline. The unknown component model for the MS subtask is a character-level MEMM which predicts whether each character in the token is the beginning or is inside of the prefix, stem, or suffix segment of the token; further details are discussed later. The STLL subtask's unknown component model is a MaxEnt model which uses Morfette-like edit scripts to change the stem-lemma string transduction task to a stem-lemma transduction pattern "classification" task (see Section 6.3.2 for more details). The HHD subtask's unknown component model uses the same heuristic as in the baseline to predict that the lemma should be added to the dictionary as the first entry with that headword. The unknown component MaxEnt models for the MS and STLL subtasks can be trained on fewer than all of the annotated examples but still maintain reasonable accuracy by training with only the first occurrence of each input-label type pair (see the results section for validating experiments).

The MS subtask's unknown component model is a composite word and character MEMM model which uses a begin/inside tagging (BI-tag) scheme. Composition of a character-level MEMM in a word-level MEMM was done by Haertel et al. (2010b), which in fact inspired the hybrid design of all of the other subtask models in SyroMorph (McClanahan et al., 2010). Hence, the character-level model has the potential to use features from surrounding words and their labels; however, so far we have not found such context features to be useful so we only use features from the current word token. The feature templates we use in the character-level MEMM are summarized in Table 7.2. The CHARACTER template extracts individual characters at the current position (the current character) and the previous and next five positions; special start and end delimiters are used for offsets beyond the bounds of the word. The PREVNGRAM and

NEXTNGRAM templates extract character n-gram combinations of size two to five strictly preceding and following (respectively) the current position; special start and end delimiters are used for offsets beyond the bounds of the word. The PREFIXCHAR template indicates whether a character is known to be a common prefix character of the language being processed; this is the only feature template among all of the features used in all of the subtasks which includes a some domain knowledge. The PREVLABEL and TWOPREVLABELS templates extract the previous and two previous (respectively) BI-tags; a special start symbol is used for previous offsets beyond the bounds of the word. The inventory of BI-tags includes one begin tag and one inside tag per prefix, stem, and suffix segment part type: {*B-PRE*, *I-PRE*, *B-STE*, *I-STE*, *B-SUF*, *I-SUF*}. The POSSIBLETAG and IMPOSSIBLETAG templates extract all of the possible and impossible (respectively) BI-tags for the current position given the previous tag and the distance from the end of the word. The BI-tags are constrained to require a stem and to allow at most one prefix, stem, and suffix segment in that order. The STARTOFWORD and ENDOFWORD templates extract the distance of the current character from the start and end (respectively) of the word if the distance is less than 4.

Name	**Template**	**Example**
CHARACTER	(±*offset*, *character*)	{ (-5, *?*), (-4, *?*), (-3, *?*), (-2, <), (-1, *i*), (+0, *y*) (+1, ~), (+2, *a*), (+3, *A*), (+4, *k*) (+5, *a*) }
PREVNGRAM	[*char*, *char*, ⋯, *char*]	{ [*?*, *?*, *?*, <, *i*], [*?*, *?*, <, *i*], [*?*, <, *i*], [<, *i*] }
NEXTNGRAM	[*char*, *char*, ⋯, *char*]	{ [~, *a*], [~, *a*, *A*], [~, *a*, *A*, *k*], [~, *a*, *A*, *k*, *a*] }
PREFIXCHAR	*true* or *false*	*false*
TWOPREVLABELS	*[BI-tag, BI-tag]*	[*B-STE*, *I-STE*]
PREVLABEL	*BI-tag*	*I-STE*
POSSIBLETAG	*BI-tag*	{ *I-STE*, *B-SUF* }
IMPOSSIBLETAG	*BI-tag*	{ *B-PRE*, *I-PRE*, *B-STE*, *I-SUF* }
STARTOFWORD	*distance from start of word*	3
ENDOFWORD	*distance from end of word*	*not fired*

Table 7.2: Feature templates provided in our implementation for the unknown MS component maximum entropy model. *BI-tag* ∈ {*B-PRE*, *I-PRE*, *B-STE*, *I-STE*, *B-SUF*, *I-SUF*} Examples are for stem 1:5:1 <*iy~aAka* character 3 (*y*) in Chapter 1 of the Quran (see Table 5.1). Five preceding and following characters are extracted, where *?* indicates an index beyond the bounds of the word. N-grams of size two to five are extracted which strictly precede or follow the current character, where *?* indicates an index beyond the bounds of the word. Possible and impossible tags are given assuming the previous tag is *I-STE*. STARTOFWORD and ENDOFWORD only fire if the distance from the start or end of the word respectively is less than 4.

Each subtask's known ambiguous component model is a set of MaxEnt models, one model for each input type known to have two or more labels and which is trained on all of the labeled instances of the input type. Each known ambiguous MaxEnt model extracts a variety of lexical, position, and previ-

ous label features, as summarized in Table 7.3. The OFFSETPREFIX and OFFSETSUFFIX features extract greedy prefix and suffix character sub-sequences respectively. Greedy affixes of every length from 1 to $min(4, input.length - 1)$ are extracted from the three preceding, current, and three succeeding input instances. The MS and HHD subtasks always extract the greedy affix features from the input instances, whereas the STLL subtask extracts the greedy affix features from preceding lemmas (tags) when available, but otherwise extracts them from the input instances.[2] The STARTOFSEQUENCE and ENDOFSEQUENCE templates extract the distance of the current input from the beginning and end of the sequence respectively if the distance is less than three. The NEARBEG and NEAREND templates respectively fire when the STARTOFSEQUENCE and ENDOFSEQUENCE templates do. The STARTOFSEQUENCE and ENDOFSEQUENCE templates provide a quantitative measurement of the current input from the beginning or end of the sentence, whereas the NEARBEG and NEAREND templates provide a qualitative indication of whether the current input is close to one of the ends. The PREVLABEL template extracts the previous label when present. Similarly, the TWOPREVLABELS template extracts the two previous labels when both are present. Only the HHD subtask employs the PREVLABEL and TWOPREVLABELS features.

7.3 Incremental Training Algorithm

In this section we present the training algorithm we employ to train the MEMM model incrementally and with partially annotated data. We employ incremental training algorithms for the CDL pipelines so as to achieve a balance between high accuracy and quick, interactively suitable training times. In Section 7.3.1 we summarize how we use terms such as "interactive", "online", and "batch". We identify our algorithm initialization and tuning choices in Section 7.3.2. Then, in Section 7.3.3, we explore ways in which to learn from partially annotated data by interpolating missing tags but we ultimately use a simpler approach amenable to the MEMM model which simply learns from the current state of the annotated data.

7.3.1 Terminology

We make a distinction between how the model is initialized when new training data becomes available and how the model is tuned given the available training data. Initialization may be done *from scratch* by starting

[2]For the MS subtask in previous versions of SyroMorph, the greedy affix templates did extract from previous labels (segmentation patterns), but with a limit of size four the only greedy suffixes that were extracted would simply indicate the presence of a suffix ("@SUF") or stem ("@STE") but no information about the content of the suffix or stem because of the segmentation representation used: for example, *l*@PRE*mlk*@STE*kwn*@SUF is the segmentation pattern for the Syriac word *lmlkkwn*.

Name	Template	Example
OFFSETPREFIX	(±*offset*, *prefix string*)	{ (-2, <), …, (-2, <*iy*~), (-1, *E*), …, (-1, *Eaba*), (0, <), …, (0, <*iy*~), (+1, *n*, …, (+1, *naso*) }
OFFSETSUFFIX	(±*offset*, *suffix string*)	{ (-2, *A*), …, (-2, *y*~*aA*), (-1, *a*), …, (-1, *bada*), (0, *a*), …, (0, *aAka*), (+1, *u*, …, (+1, *iynu*) }
STARTOFSEQUENCE	*distance from start*	2
NEARBEG	*1 or 0*	1
ENDOFSEQUENCE	*distance from end*	1
NEAREND	*1 or 0*	1
HHD Task Only		
PREVLABEL	*label*	"<*iy aA* (2)"
TWOPREVLABELS	*[label, label]*	["<*iy aA* (2)", "*Eabada* (1)"]

Table 7.3: Feature templates provided in our implementation for known ambiguous maximum entropy models. Examples are for stem 1:5:3 <*iy*~*aAka* in Chapter 1 of the Quran (see Table 5.1). The examples use a maximum offset spans of length 3 and maximum prefix and suffix lengths of 4. Note that the prefixes and suffixes with negative offset from the current stem are derived from the lemmas whereas those with zero or positive offsets are derived from the stems. The labels for the HHD task are dictionary entry keys which include the lemma and a numeric identifier; the identifiers used are for demonstration and are not derived from actual data.

model parameters back at default values whenever the training data changes, or initialization can be done *incrementally* by using the parameters of the model instance which existed before new training data became available. Tuning may be done in *batch*, where each adjustment in the parameters is informed by all of the training data, or can be done *online*, where parameters are adjusted given one training example at a time. Other tuning strategies are possible such as mini batches and "online-to-batch" (e.g., Sokolov et al., 2015) tuning which applies online updates in epochs over all of the available training data and can thus be characterized as a batch tuning algorithm at the granularity of epochs. Online training algorithms lend themselves well to incremental training, especially for "infinite" streams of data, but online-to-batch training over a finite set could also initialize parameters from scratch as new training data becomes available. Thus, for our purposes the terms *incremental* and *online* are not synonymous and we employ an incremental batch training algorithm without contradiction.

7.3.2 Algorithm Initialization and Tuning

Training the MEMM incrementally is a simple adaptation of regular batch training which simply continues training from the last set of feature weights when new training data becomes available. The baseline pipeline, by the nature of the empirical statistics and heuristics used, is trivial to train from scratch or incrementally: simply collect or update empirical counts respectively. MEMM models, on the other hand, are traditionally trained from scratch with a batch tuning algorithm over a fixed amount of training data. Retraining the MEMM from scratch on each incrementally larger batch of annotated data is impractical, thus the development of a practical incremental training algorithm for the MEMM is a key contribution of this thesis. Generally, incremental training works by using the parameter values of the previous instance of the model to initialize the parameter values of the new model instance, and then the model parameters are tuned on the annotated training data. In the case of the MEMM, there are two incremental initialization methods we considered for the model's feature weights: (1) **continuation**: the feature weights are initialized to their previous values, or (2) **regularization**: the feature weights are all set to zero but then regularized using the previous feature wieghts as mean values of Gaussian priors (Chelba and Acero, 2004). In either case, the initial weight for any new features extracted from new annotated data is zero, the same as the implicit previous feature weight. We chose to use the incremental continuation initialization strategy and to tune parameters in batch with L-BFGS optimization of the standard MaxEnt stochastic gradient objective for several reasons. First, it was quicker to implement than regularization initialization. Second, and more important, given the intuition that the previous feature weight is in fact already close to the value that the optimization algorithm will set it to, it seems less wasteful to start at the target weight instead of starting at zero and re-traversing the gradient. Granted, the assumption of initial and final feature weight proximity may not hold as well early on in an annotation project as compared to later in the project, and so it will be valuable future work to compare the continuation and regularization initialization approaches.

7.3.3 Learning from Partially Annotated Sequences

To enable high-quality, interactive sequence tagging pre-annotation, the training algorithm needs to be able to learn from partially annotated sequences so as to train the model given all of the available annotations. Partial sequence annotations may be obtained from (simulated) human annotators as in our case (also, e.g., Kristjansson et al., 2004), or be projected from annotated data in a different language (e.g., Täckström

et al., 2013), or be projected from annotated data in a different domain of the same or related language (e.g., Tsuboi et al., 2008), and so on. There are several ways in which model training algorithms for fully annotated sequences, including the MaxEnt training algorithm for the MEMM, could be adapted to interpolate tags for the unannotated tokens in partially annotated sequences. We review several adaptation options, including bootstrapping, evidence-consistent marginalization, and evidence-consistent enumeration, but we ultimately use a simpler, pragmatic approach that only uses the features current in the training data and does not changing the standard conditional log-likelihood MaxEnt training objective and gradient. The notation used in the example figures is summarized in Table 7.4.

Symbol	Meaning
i	general position in the sequence; previous and suceeding positions are indicated by an offset: $i-1$, $i+1$, and so on
w_i	word token at position i
t_i	tag at position i
v_i	one possible tag value for the tag at position i
v'_i	another possible tag value; and so on for further prime ticks
shading	observed value
dashes	unannotated and interpolated
box	for each position in the sequence the MEMM may extract features from the whole input sequence which is represented by the vertical arrows being drawn from the box around the input sequence to each of the tag nodes

Table 7.4: Notation used in the example figures, Figures 7.2–7.5.

Bootstrapping

A simple adaptation to handle partially annotated sequences in the MEMM is to bootstrap and use the previous model instance to fill in the missing annotations. A bootstrapping algorithm could proceed by first using the current model to predict the most probable tags for the unannnotated tokens, second tuning model parameters on the now fully "annotated" sequence, and third repeating until some stopping criterion is met. For example, as in Figure 7.2, say the i^{th} word in a sequence is annotated with tag value v'_i, then bootstrapping would predict the most probable values for the missing tags for the rest of the sequence both before and after the i^{th} position and thus produce a fully annotated sequence with which to train. Bootstrapping low quality models deteriorates prediction quality and thus, without a proper confidence threshold to control which sequences to train with (cf. Tomanek and Hahn, 2009), may not be the best option when starting a fresh annotation project. A further disadvantage to bootstrapping is that the time to predict tags may be non-

negligible when there are many partially annotated sequences, such as in the experiment scenarios using the descending or ascending word type frequency data selection strategies.

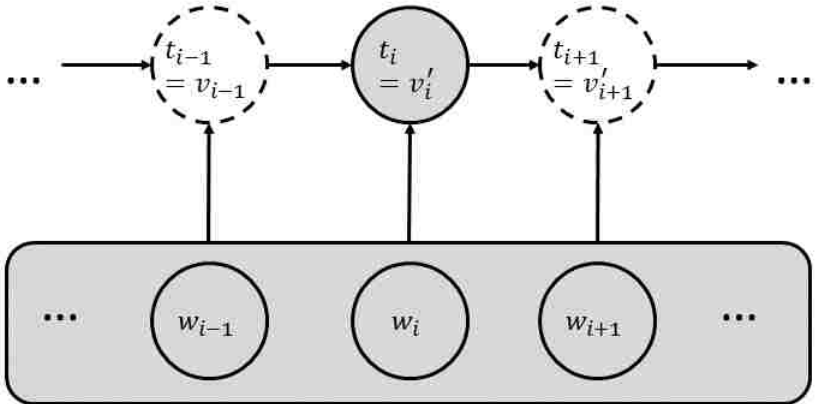


Figure 7.2: MEMM bootstrapping approach to learn from partially annotated sequences. See Table 7.4 for an explanation of the notation.

Evidence-Consistent Marginalization

Another option to handle partially annotated sequences in the MEMM is evidence-consistent marginalization which changes the training objective function so as to marginalize over all of the possible tag sequences which are consistent with the current evidence, that is the partial annotations and tag dictionary constraints. Evidence-consistent marginalization is the term we use to describe the approach taken by Tsuboi et al. (2008), Truyen et al. (2008), and Täckström et al. (2013) to train CRF sequence tagging models with partially annotated data; Truyen et al. (2008) also use the same method to train MEMMs with partially annotated data. All of those research groups define a modified conditional log-likelihood objective function which marginalize over all of the possible tag sequences which are consistent with the current evidence. The set of evidence-consistent taggings can be represented as a lattice where (1) each path represents a tagging, (2) all paths go through the single-option partial annotations, and (3) there is one path for each combination of applicable tags for the unannotated tokens in the sequence. For example, in Figure 7.3 there are different sets of applicable tags for each token, the i^{th} token has been annotated with tag v'_i and the evidence-consistent paths include $< \cdots, t_{i-1} = v_{i-1}, t_i = v'_i, t_{i+1} = v'_{i+1}, \cdots >, < \cdots, t_{i-1} = v'_{i-1}, t_i = v'_i, t_{i+1} = v''_{i+1}, \cdots >$ and so on, but do not include paths where $t_i \neq v'_i$. Evidence-consistent marginalization is appealing since it interpolates missing tags without requiring sequence decoding as with bootstrapping. With the modified objective function, training then proceeds by using stochastic gradient optimization using the gradient of the

new objective function. Although the evidence-consistent marginalization approach can be applied to the MEMM with reasonble results (Truyen et al., 2008) the likely time to implement, validate, and apply the method to our research was beyond the time constraints of this work.

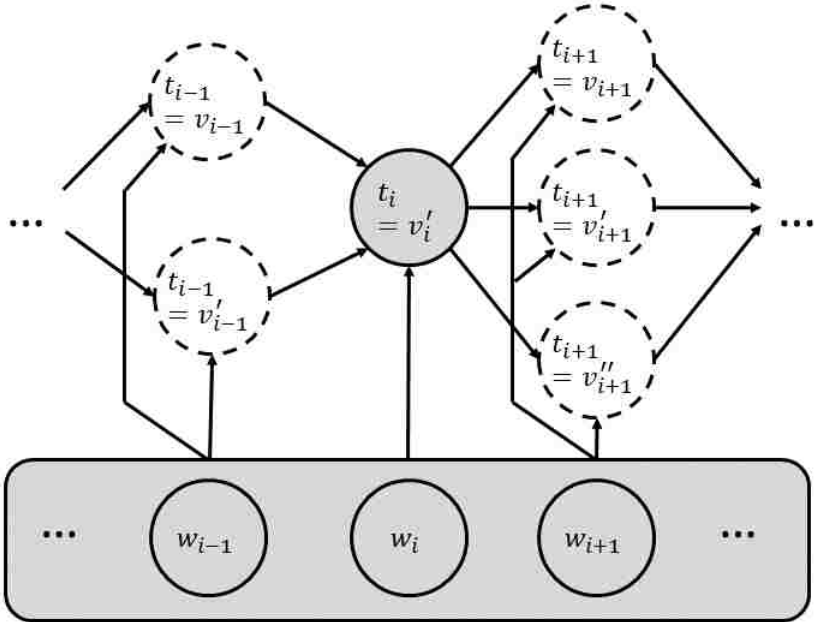


Figure 7.3: MEMM evidence-consistent marginalization and enumeration approaches to learn from partially annotated sequences. The lattice of annotated and interpolated tags represents the possible tag sequences which are enumerated or marginalized over. See Table 7.4 for an explanation of the notation.

Evidence-Consistent Enumeration

Another adaptation to handle partially annotated sequences in the MEMM is simply to enumerate all of the taggings consistent with the partial annotations and tag dictionary evidence and thus get training data which is fully annotated. The set of consistent taggings can again be represented as a lattice as in Figure 7.3. However, the enumerated data would also be highly contradictory since for each unannotated token there would be one training instance with every possible combination of consistent tag and previous consistent tag or tags depending on the Markov order. Granted, the conflicting training data may serve to keep the model regularized but it also bloats the training data and thus unnecessarily increases the time to compute the gradient. Given the MEMM's locally normalized structure, to obtain valid probabilities it is not necessary to train the local MaxEnt model with all of the unannotated tokens with interpolated tags: instead, the model can just be trained with the annotated tokens and use the interpolated tags to provide Markov (previous tag)

features. Indeed, we take this a step further and do not interpolate missing tags at all.

Existential Feature Extraction (Our Approach)

To train the MEMM with partially annotated data, we simply train the local MaxEnt model with the features that currently exist in the annotated data. For example, as shown in Figure 7.4 missing tags are not interpolated at all. If an annotated token's preceding tokens are also annotated then the token can have Markov features in its feature vector, as in Figure 7.4, but otherwise the feature vector does not currently include Markov features, as would be the case in Figure 7.4 if the token at position $i-1$ was not annotated. Over time, as more data is annotated, then the feature vector representation for each annotated token grows to include Markov features when its neighbors are annotated. This pragmatic, existential approach thus trains the model with the data that is available without interpolating extra data. At the outset of a fresh project with no or little annotated data, the MEMM behaves like a MaxEnt classifier since the Markov structure is dormant (not utilized), but over time the model adapts to leverage the Markov potential insofar as the annotated data supports it.

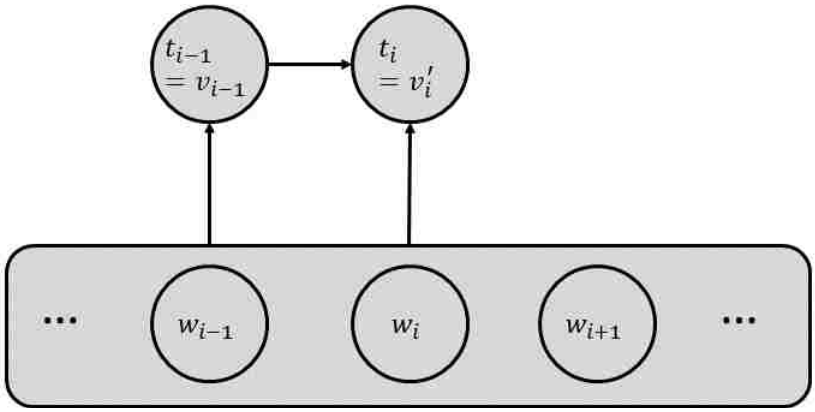


Figure 7.4: MEMM existential feature extraction approach to learn from partially annotated sequences. The feature vector for the i^{th} word token can include features from the preceding tag, whereas the feature vector for the word token at position $i-1$ does not currently include Markov features. See Table 7.4 for an explanation of the notation.

In addition to using only "real" training data, and requiring no recoding of the MaxEnt objective function, existential feature extraction has several other pragmatic benefits. First, in the hybrid MEMM model, the unknown component MaxEnt model can be trained on fewer examples but still maintain reasonable accuracy by only training on select rare input types. Selecting a subset of the annotated data to train with is easier to do using existential feature extraction and regular MaxEnt training, instead of evidence-consistent

marginalization, which pull other non-select input types into the objective function, or evidence-consistent enumeration, which bloats the data with conflicting Markov features. Second, the STLL and HHD components in the hybrid MEMM pipeline only extract features from the stems and lemmas, or lemmas and dictionary keys respectively. This means that the training data for the STLL and HHD components has partial input sequences as well as partial output sequences; for example, the middle layer in Figure 7.5 is the input to the top layer but is only partially defined. Yet, using the existential feature vector which adapts and expands during the course of annotation, the training algorithm is able to accommodate partial input sequences just as well as partial output sequences. Another option to deal with partial input sequences, and to possibly boost accuracy, would be to utilize features extracted from the token sequence in all of the layers of the pipeline; however, such feature engineering is not necessary to demonstrate that our incremental training algorithm is suitable for interactivity so we leave the feature engineering to future work. Finally, training with the existential features was simple to implement, is likely the fastest of the identified partial annotation training algorithm options, and thus establishes results for incremental, interactive training which further future exploration of the other training algorithms can compare against.

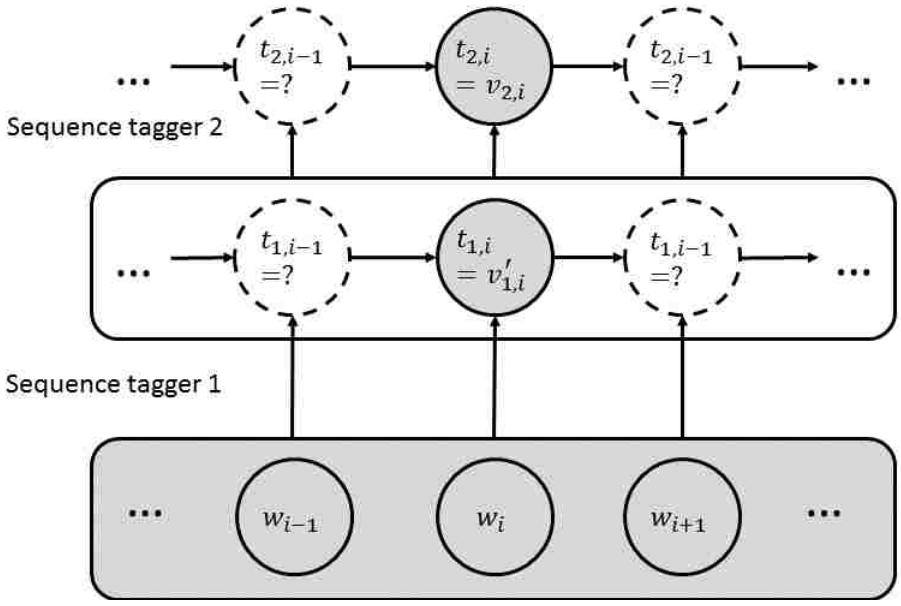


Figure 7.5: Pipeline of 2 MEMMs. The bottom layer is the input and the middle layer is the output of the first tagger, and the middle layer is the input and the top layer is the output of the second tagger. The taggers can easily learn from both partial input and partial output sequences using existential feature extraction since missing values need not be interpolated as with other approaches for learning from partially annotated sequences. See Table 7.4 for an explanation of the notation.

7.4 Experimental Design

This section presents the experimental design to validate that the hybrid pipeline is suitable to provide interactive pre-annotation assistance using the incremental training algorithm. In particular we discuss the accuracy and training time measures we use to analyze model performance.

For each experiment scenario we start with an unlabeled corpus and continue the annotation simulation until the whole corpus has been annotated. We simulate each scenario on the four data sets PNT/D, PNT/U, QC/D, and QC/U as summarized in Table 7.5. The development data sets are treated as the corpora of interest to be interactively annotated. The blind test sets are held out such that instances in those sets are never used to train the models. We simulate an oracle annotator by using the gold-standard labels from the annotated data set; the simulated annotator thus has access to the part of speech information which the annotator uses to consistently disambiguate among dictionary entries with the same headword.

	All			**Development** (80% of All)			**Blind Test** (20% of All)		
Source Data	verses	tokens	types	verses	tokens	types	verses	tokens	types
Arabic QC/D	6236	77,429	18993	4988	61,957	16558	1248	15,472	6212
Arabic QC/U			14880			13103			5145
Syriac PNT/D	7957	109,640	18359	6365	87,560	16199	1592	22,080	6868
Syriac PNT/U			16439			14573			6340

Table 7.5: All, development, and blind (held-out) test set partitions used throughout this chapter to analyze ambiguity, develop models, and evaluate model accuracy. The columns labeled "tokens" and "types" are word tokens and word types. The numbers for "All" are repeated from Table 5.3 for convenience.

To assess the usefulness and benefit of pre-annotation assistance we record and analyze held-out and progressive validation accuracy measures. Held-out accuracy for a model is measured by applying the model to predict annotations for a held-out labeled test set (the blind test set) and then comparing the agreement between the predicted labels and gold-standard labels. We measure held-out accuracy for every experiment scenario in all of the evaluation configurations. Measuring held-out accuracy on thousands of verses and tens of thousands of tokens takes several minutes so we only measure held-out accuracy at select points during each simulation. Specifically, we measure held-out accuracy before any model updates, after the very last model update, at the model update points that correspond to the 10% increment marks relative to the total number of model updates, and after model updates $\{1, 2, 5\}$ times each applicable order of magnitude. Progressive validation accuracy (Blum et al., 1999), or online cumulative accuracy, is the running ratio of

correct decisions to all decisions. We measure progressive validation accuracy for evaluation configurations 2 and 3 which simulate complete annotation of the corpus. A decision is deemed correct or not in hindsight after the annotator provides an annotation, that is, accepts or corrects the label. Progressive validation estimates of a model's performance are guaranteed to have the same error bounds as held-out estimates obtained on data sets of the same size (Blum et al., 1999). Given the relative sizes of the development and held-out data sets, in the annotation simulations the progressive validation accuracy is evaluated on more data than the held-out accuracy for approximately the last three quarters of the learning curves, and so, because of volume, is a less-variable indicator of model performance than is the held-out accuracy, assuming that data continues to be selected for annotation with the same data selection strategy used so far. The progressive validation accuracy is a measure of how much (or little) effort the annotator must have invested to correct decisions over time and therefore is an indication of how the annotator likely perceives the helpfulness of the pre-annotation assistance. Pre-annotations are updated in the simulation algorithm (line 11 in Algorithm 1) using correction propagation (CP) (Kristjansson et al., 2004) such that the model provides pre-annotation suggestions for the sequence which include and are influenced by any existing annotations for the tokens in the sequence; given a new token annotation, the model need not be retrained before updated and improved pre-annotations for the sequence can be provided.

In our analysis, we report average accuracy instead of showing a learning curve when appropriate. We compute the average accuracy by calculating the area under the accuracy learning curve and then dividing by the total potential area. When calculating the average accuracy broken down by ambiguity cases—unknown, known unambiguous, and known ambiguous—we scale the accuracy area and total potential area by the percentage share of the test data for the ambiguity case. Scaling by the percentage share of the ambiguity case avoids overestimating the average accuracy as happens if the accuracy is not scaled.

To assess the costs of pre-annotation assistance and current interactive suitability we measure the real time it takes to train the pipeline models. The total cost to have humans annotate a corpus includes model training time and the time it takes the annotators to validate and correct pre-annotations Haertel (2013). If model training is done in parallel to providing pre-annotations and the training time is shorter than the time it takes annotators to validate and correct pre-annotations, then the pre-annotation assistance will likely be perceived as interactive and the dominant cost will be the pace at which the humans work. We do not have empirical data with which to build a model for human CDL annotation time that matches our CDL simulation setup so instead we content ourselves with just measuring the model training time. The evaluation

configuration 1 experiments measure batch training time since the batch training algorithm is applied from scratch at each snapshot point. The evaluation configuration 2 and 3 experiments measure incremental training time since the incremental training algorithm is used for each model update. Granted, software and hardware engineering dramatically impact speed, so training time is not a perfect surrogate for algorithm complexity and hence for determining whether the algorithm is universally suitable for interactivity. Yet, putting aside the issue of measuring inherent complexity, we use the real training time to answer the practical question of whether our model implementation seems good enough as it is to employ for interactivity on current modern hardware. Each machine in our processing cluster has an 8 logical core, 3.50GHz CPU (Intel® Core™ i7-4770K) and 32 GB of RAM.

We aim to have model training times that are 10 seconds or less so that the pre-annotation assistance can reasonably be deemed as being interactive. Notwithstanding that we do not have empirical data that matches our CDL simulation setup, we conjecture that the time it would take an annotator to annotate a token with segmentation, lemma, and dictionary entry would be 10–30 seconds which is comparable to the per-token time it takes human annotators in other other language annotation tasks. For example, annotators working on part of speech tagging took 14–20 seconds per token (Ringger et al., 2008) and annotators working on Syriac morphological tagging took 16–30 seconds per token on just segmentation and stem-to-lemma linkage (data collected in user study conducted by Felt et al., 2013). Thus, in our analysis we aim to have model training times which are 10 seconds or less so that the training time would likely not be the dominant annotation cost in a real annotation project. In a real annotation project, especially with a 10 second training time, model training and pre-annotation prediction should be done in parallel, but because we are just simulating annotation we train and predict in sequence.

7.5 Results and Analysis

In this section we analyze the performance of the hybrid pipeline compared to the baseline pipeline. We look at the accuracy and training time of the models in each of the annotation simulation scenarios summarized in Table 7.1. We first present parameterization experiments in Section 7.5.1 to determine the parameters to use for the hybrid pipeline's unknown component models. Then, in Section 7.5.2, we analyze CDL annotation conducted using a reading order data selection strategy and demonstrate that incremental training is interactively viable. Given that the model can be trained quickly with partially annotated sequences,

we then explore in Section 7.5.3 other ways of traversing the corpus using word frequency data selection strategies and analyze the affect on accuracy and training time compared to the reading order traversal. The terminology used throughout this section is summarized in Table 7.6, some of which has been explained in previous sections and the remainder of which will be explained in the analysis.

7.5.1 Parameterization Experiments

In order to be suitable for interactive pre-annotation assistance, the hybrid pipeline's unknown component models need to be trained quickly but also maintain high accuracy. We look at two parameters for training the MEMM unknown component models: the training input filter and the stopping criteria for L-BFGS optimization. For evaluation configuration 1, which takes snapshots of the model performance, the hybrid will be trained on all of the data and L-BFGS optimization will be allowed to run to convergence. The remainder of this section focuses on choosing parameters for evaluation configurations 2 and 3 where complete annotation of the whole corpus is simulated and the models are trained incrementally after each sequence or token is annotated or all instances of a word type are annotated. The experiments are conducted on the development portion of the Peshitta New Testament: 75% is used as the simulation corpus (training data), and 25% is held out as a development test set. In later sections the chosen parameterization will be used for all data sets to simulate annotation of the whole development set. Sequences are selected at random using the reading order data selection strategy and also to determine the order in which word tokens of a selected word type are annotated when using the descending or ascending word type frequency data selection strategies.

We explore a variety of training input filter and stopping criteria combinations. We evaluated the following training input filters:

- ***all***: train with all of the available data,
- ***singletons***: train with just the task input types that currently have one annotated instance; once an input type has two or more annotated instances it is taken out of the training set
- ***first input-label pairs***: train with the first observed instance of each pair of input type and label (tag) type; unlike the *singletons* filter, the *first input-label pairs* filter does not remove data from the training set over time.

We experimented with the following stopping criteria for L-BFGS optimization:

- ***convergence***: run until the change in the negative conditional log likelihood (NCLL) score is less than $1e^{-6}$

Group	Term	Meaning
Data Selection Strategies		
	RO	sequences selected in natural or random reading order and tokens annotated in order within the sequence
	DWF	word types selected by descending frequency; tokens per word type iterated in natural reading order
	AWF	word types selected by ascending frequency; tokens per word type iterated in natural reading order
Model Update Strategies		
	SEQ	model updated when the sequence changes
	TOK	model updated when the token changes
	WTC	model updated when the word type changes
Simulation Mode		
	Snapshot	train and evaluate models with set amounts of data in parallel
	Complete	train and evaluate models sequentially, thus simulating the complete annotation project
Data Sets		
	QC	data is the Quranic Corpus (Arabic)
	PNT	data is the Peshitta New Testament (Syriac)
	/U	data includes no diacritics
	/D	data includes all available diacritics
Models		
	baseline (B)	CDL pipeline uses heuristics and memorizer
	hybrid (H)	CDL pipeline uses hybrid MEMM models
MEMM Input Filters		
	all	MEMM trained with all annotated data
	first input-label pairs (filps)	MEMM trained with first observed instance of each pair of input type and label type
MEMM Stopping Criteria		
	convergence	MEMM trained until the change in objective function is less than $1e^{-6}$
	max 10	MEMM trained until converges or for at most 10 iterations, whichever comes first
Accuracy Measures		
	ho	held-out accuracy
	pv	progressive validation accuracy

Table 7.6: Terminology Reference. Commonly used terminology for interpreting the results presented in Section 7.5. The table is not comprehensive: section-specific terminology is explained in the appropriate section.

- ***max 10***: allow up to 10 iterations of optimization but stop if the change in NCLL is less than $1e^{-6}$.

Again, for evaluation configuration 1 the training input filter is *all* and the stopping criteria is *convergence*. For configurations 2 and 3 we evaluate the cross product of input filters and stopping criteria.

The parameter combination which best balances accuracy and incremental training time is to use the *first input-label pairs* and *max 10* stopping criteria. To achieve the best generalization accuracy it would be desirable to use the *all* filter and *convergence* stopping criteria (as used for evaluation configuration 1), but in our pilot experiment with those parameters for the reading order and sequence change scenario (RO+SEQ) the incremental training time is consistently longer than 10 seconds and towards the end of the simulation is close to three minutes (180 seconds). The training time is still consistently longer than 10 seconds with the *all* filter and *max 10* stopping criteria. The *singletons* and *first input-label pairs* filters have better training times and do not noticeably degrade prediction accuracy compared to the *all* filter. Using the *singletons* and *first input-label pairs* filters with either the *convergence* or *max 10* stopping criteria, the unknown component models achieve about the same progressive validation accuracy in the reading order scenarios (RO+SEQ and RO+TOK); for example, see the progressive validation unknown accuracy for RO+TOK in Figure 7.6a (the legend is in Figure 7.6f). However, in the descending word frequency scenarios (DWF+TOK and DWF+WTC) there is a large difference in performance. In the DWF+TOK scenario, shown in Figure 7.6c, with between 10,000 and 30,000 annotated tokens, the *first input-label pairs* filter yields slightly lower accuracy than does the *singletons* filter and the baseline, but more concerning is that towards the end of the simulation when the baseline starts to drop that the *singletons* filter also drops and dips below the baseline. The dip is even more dramatic in terms of held-out accuracy, as shown in Figure 7.6e. The reason for the dip is that the *singletons* filter causes model training to restart from scratch as soon as the second instance of each word type is encountered using the descending word type frequency data selection strategy, such that by 50,000 annotated tokens, the unknown component model is just barely beginning to be trained and a stable set of singleton data. Furthermore, the fact that the accuracy dips indicates that there is a lot of diversity in the singleton data types and that the model initially over fits to special cases until it has accumulated enough data to start regularizing and pick accuracy back up. The *first input-label pairs* filter does not have the same dip in accuracy as the *singletons* filter because it does not discard models during the course of the simulation, such that by 50,000 annotated tokens, the model is regularized and is able to consistently improve with the addition of new special case training inputs. In terms of accuracy the *convergence* and *max 10* stopping criteria are about the same, but in terms of training time the *convergence* stopping criteria

is more expensive and approaches and exceeds the 10 second interactive-perception maximum mark sooner; for example, see the training time curves for RO+TOK in Figure 7.6b and for DWF+TOK in Figure 7.6d. Thus, the parameters we use in later complete simulation experiments (evaluation configurations 2 and 3) are the *first input-label pairs* filter and *max 10* stopping criteria.

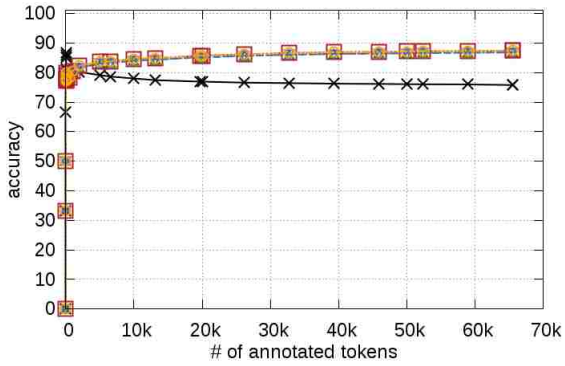
7.5.2 Reading Order Selection Strategy Experiments

In this section we compare incremental training to traditional batch training by analyzing the experiment scenarios which use the reading order data selection strategy. The experiment scenarios include the evaluation configuration 1 reading order and sequence change snapshot simulation (snapshot RO+SEQ), configuration 2 reading order and sequence change complete simulation (complete RO+SEQ), and configuration 3 reading order and token change complete simulation (RO+TOK). The hybrid pipeline's unknown component models use the *all* filter and *convergence* stopping criteria for the snapshot RO+SEQ scenario and the *first input-label pairs* filter and *max 10* stopping criteria for the complete RO+SEQ and RO+TOK scenarios. We compare model performance between two reading order selection strategies: the first where the training set is assembled by selecting sequences (verses) in the natural corpus order, and the other using a random reading order—select sequences at random and iterate the tokens in reading order—without replacement such that the last training set in all experiments is the full development set.

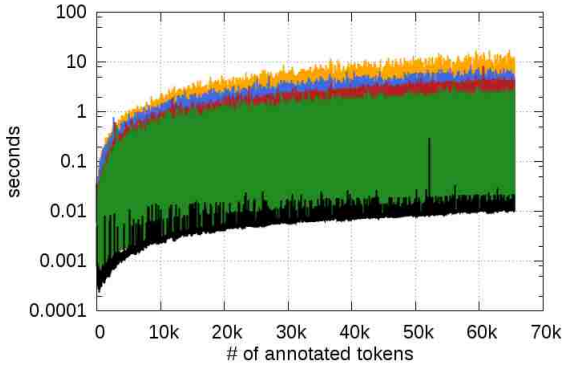
There are four principal conclusions from the analysis. (1) Novel tokens are selected for annotation less frequently using the natural reading order as compared to a random reading order, which, intuitively, is because the naturally organized discourse is cohesive such that most of the relevant word types have already been introduced. (2) The hybrid pipeline achieves higher average held-out and progressive validation accuracy than the baseline pipeline. (3) For both pipelines, the overall average held-out and progressive validation accuracy is significantly different using the natural reading order than the random reading order. (4) The hybrid pipeline cannot be trained interactively using the traditional batch training algorithm but can be trained interactively using our incremental training algorithm.

The partition of the held-out and progressive validation test data into different ambiguity cases has similar trends but different distributions over time. At each point in the simulation, each test token can be categorized with respect to the training data as unknown, known unambiguous, or known ambiguous. The test set can thus be partitioned into unknown, known unambiguous, and known ambiguous groups. We analyze the percentage share that each group makes up in the whole test set. Figure 7.7 shows the

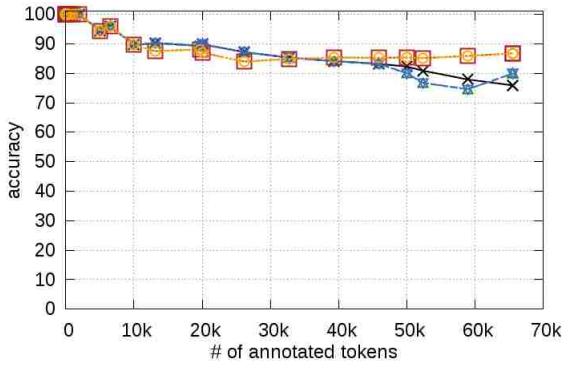
(a) RO+TOK progressive accuracy



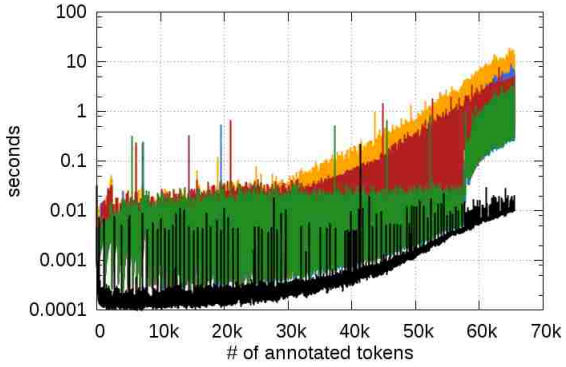
(b) RO+TOK training time



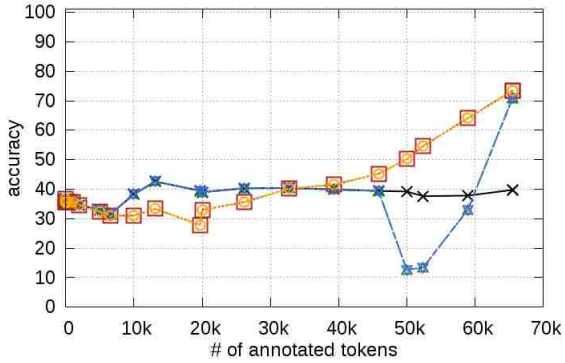
(c) DWF+TOK progressive accuracy



(d) DWF+TOK training time



(e) DWF+TOK held out accuracy



(f) Legend

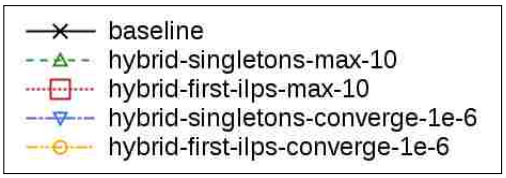


Figure 7.6: Development experiment examples for progressive validation accuracy, held-out accuracy, and training time. The training time plots (Figures 7.6b and 7.6d) only use the colors, and not the point types, from the legend (Figure 7.6f). In the legend, the term *"first-ilps"* means the *first input-label pairs* filter.

percentage share of each ambiguity case for the held-out and progressive validation test data in the complete RO+SEQ simulation using the natural reading order for each of the four data sets. At each decision point in the annotation project, or for each vertical slice, the sum of unknown, known unambiguous, and known ambiguous shares per held-out and progressive validation metric is 100%. The held-out test shares are the same for the snapshot and complete RO+SEQ scenarios and are similar for the RO+TOK scenario. The progressive validation test shares are similar for the natural RO+TOK simulation scenario and are not applicable for the snapshot RO+SEQ scenario. As expected, the unknown shares are a majority of the held-out and progressive validation test data early on in the annotation projects (left end of curves) but then gradually decrease in dominance as more of the corpus is annotated and becomes known. The known unambiguous shares become the majority shares in all data sets within the first 10,000 annotated tokens. The known ambiguous shares are larger in the Quranic Corpus than in the Peshitta New Testament, and more data is known ambiguous in both the QC and PNT when diacritics are stripped. Both the held-out and progressive validation test share distributions (vertical slices) provide estimates of the distribution of how future data will be categorized with respect to the training data. However, the progressive validation estimate is a more accurate estimate for two reasons: (1) the growing, cumulative progressive validation test set is larger than the fixed held-out test set for approximately the last three quarters of the annotation project, and (2) the progressive validation estimate accounts for the historical effects of annotating each type-label pair in the training data for the first time whereas the held-out estimate only considers how each type-label pair is currently categorized.

The rate at which novel words are selected and annotated using the natural reading order selection strategy is significantly different than using a random reading order selection strategy because the naturally organized data is cohesive and thus more likely to use words which have been used before. In Figure 7.8 we compare the natural reading order test share trends with the mean trends of ten random reading order repetitions for the PNT/D data set. The comparisons for the other three data sets are similar. For the held-out test set (Figure 7.8a), the natural reading order unknown shares are larger and the known unambiguous shares are smaller than the respective random reading order shares for much of the annotation project.[3] Conversely, for the progressive validation test set (Figures 7.8b and 7.8c), the natural reading order unknown shares are smaller and the known unambiguous shares are larger than the respective random reading order shares. The

[3]The held-out random reading order shares are interpolated to the same points at which the held-out set was evaluated using the natural reading order.

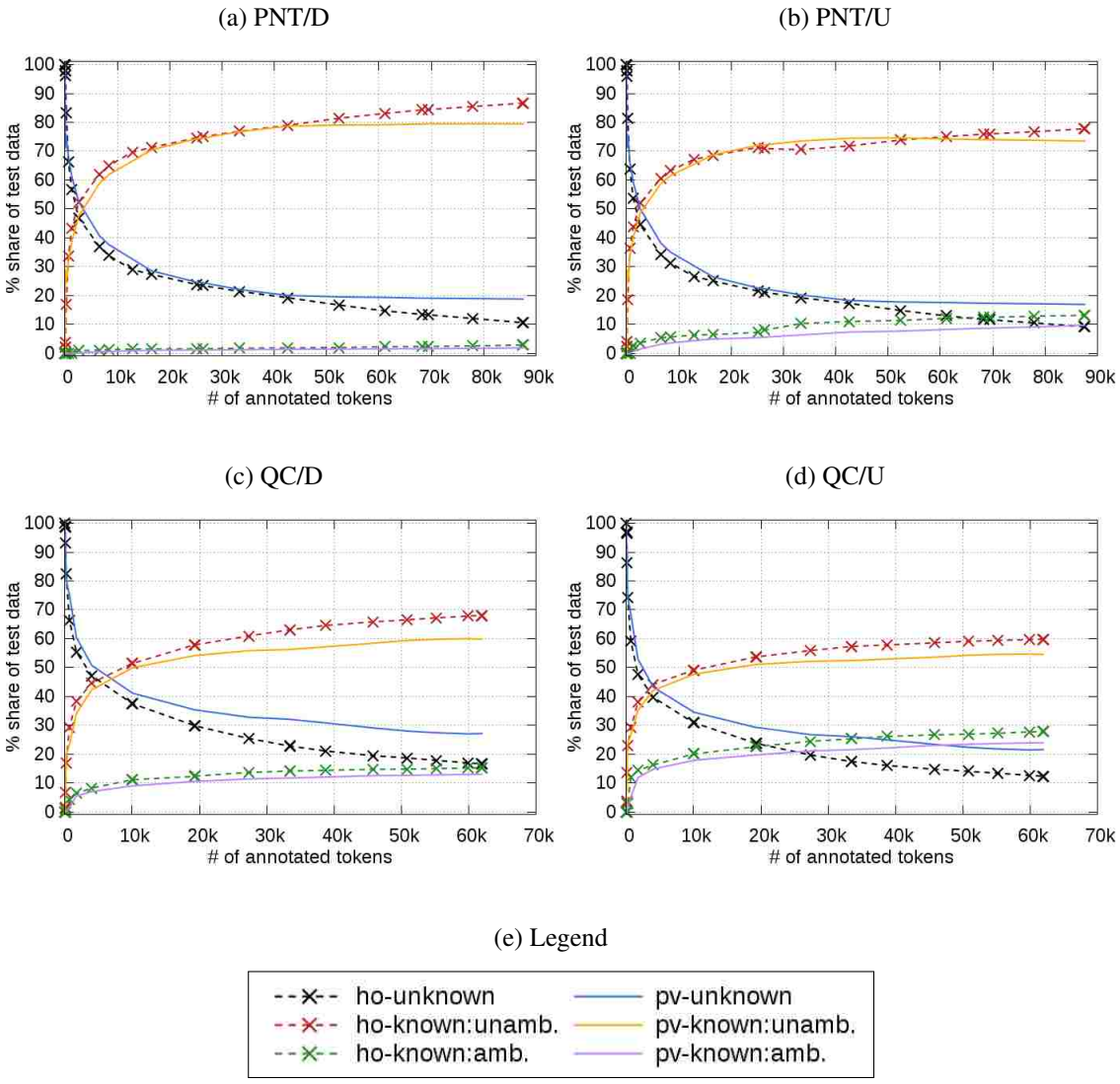
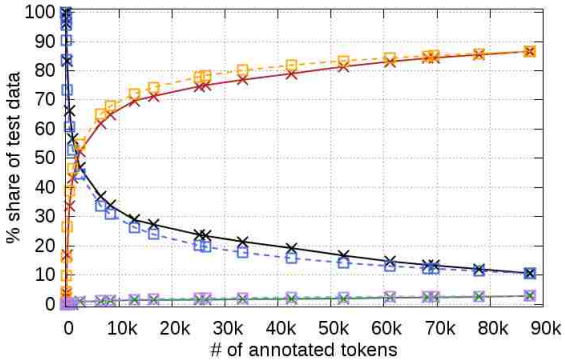


Figure 7.7: Percentage share of held-out and progressive validation test data per unknown, known unambiguous, and known ambiguous category in each of the four data sets. Test shares are computed for the complete RO+SEQ scenario (evaluation configuration 2) using the natural corpus reading order. The held-out (ho) shares are the same for the snapshot RO+SEQ scenario and are similar for the RO+TOK sceanrio. The progressive validation (pv) shares are similar for the natural RO+TOK simulation scenario.

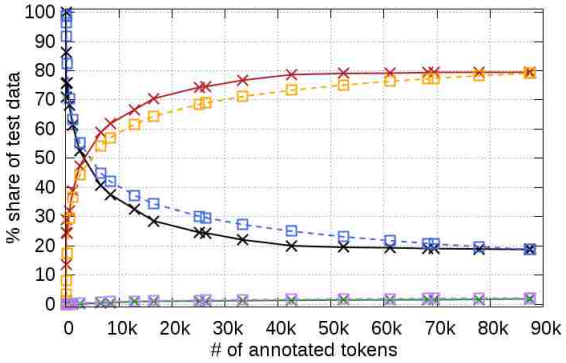
reason for the differences between natural and random reading order is the same for both kinds of test sets: the natural reading order selects novel word types for annotation less frequently than does a random reading order. This difference in the rate of novel word type annotation means that training data accumulated via the natural reading order usually has fewer annotated word types than training data accumulated via a random order. The effect on the held-out test data, comparing natural to random, is that more of the test data is categorized as unknown and less as known. The effect on the progressive validation test data, comparing natural to random, is that unknown word types are encountered less frequently than known word types. This observation has a further intuitive explanation: the naturally organized corpus is topically coherent so it is more likely to encounter a known word token relevant to the current discourse than to encounter a novel token, whereas reading verses at random introduces novel tokens more regularly. The difference between the natural and random reading orders is significant inasmuch as the standard deviations on the ten random test shares do not encompass the natural reading order test share values; the errors are so small that it would make little visual difference to display them, so the figures do not include the standard deviation.

The hybrid pipeline achieves consistently higher predictive accuracy than the baseline pipeline. To validate that claim we first analyze the accuracy learning curves from the PNT/D simulations which are representative of the trends in all of the data sets. Figure 7.9 shows the overall and unknown held-out and progressive validation accuracy learning curves using the natural reading order selection strategy on the PNT/D data set. The baseline and hybrid trends are about the same in terms of known unambiguous and known ambiguous accuracy so those curves are not shown. Visually, in terms of held-out accuracy, in the snapshot RO+SEQ scenario the hybrid model achieves and maintains higher overall and unknown accuracy than the baseline after 1,000–2,000 annotated tokens (Figures 7.9a and 7.9c respectively). By visual comparison, the held-out accuracy learning curves are about the same in the complete RO+SEQ scenario as they are in the snaphost RO+SEQ scenario (compare Figure 7.9b to 7.9a and Figure 7.9d to 7.9c); the same is true for the held-out accuracy in the RO+TOK scenario (not shown). The similarity in held-out accuracy learning curves for the PNT and QC data sets indicates that incrementally training the hybrid model, instead of training from scratch, efficiently explores the parameter space and does not need to "unlearn" and "relearn" as new data is added to the training data. Furthermore, using the more parsimonious parameterization for the unknown component—*first input-label pairs* filter and *max 10* stopping criteria—yields about the same accuracy as using the complete or generous parameterization—*all* filter and *convergence* stopping criteria. In terms of progressive validation accuracy with correction propagation, the hybrid dominates the baseline

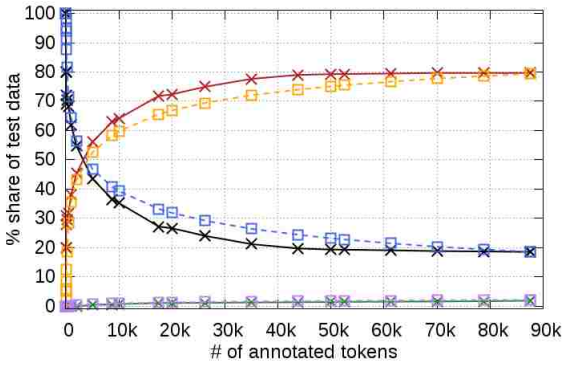
(a) PNT/D held out



(b) PNT/D RO+SEQ progressive



(c) PNT/D RO+TOK progressive



(d) Legend

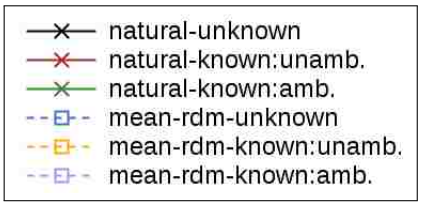


Figure 7.8: Comparison of percentage shares of unknown, known unambiguous, and known ambiguous cases using the natural reading order and mean shares from ten random reading order repetitions (*"mean-rdm"* in the legend). Test shares are shown for the PNT/D data set and are similar for the other three data sets. Held-out trends (Figure 7.8a) are the similar for all three reading order scenarios. Progressive validation trends are about the same between the RO+SEQ (Figure 7.8b) and RO+TOK (Figure 7.8c) experiment scenarios.

for the whole learning curve both in overall and unknown accuracy (Figures 7.9b and 7.9d). Table 7.7 reports the token after which the hybrid pipeline achieves higher accuracy than the baseline pipeline for the remainder of the simulation in each of the four data sets; before that point the hybrid accuracy may fluctuate around the baseline's accuracy. The hybrid dominance points quantify more precisely what was observed in the learning curves. On average, the hybrid attains dominance over the baseline will less than 700 annotated tokens across all data sets and is sometimes obtained with as little as 4 annotated tokens as with the natural reading order in the QC/D data set. Looking across all of the experiment scenarios and both held-out and progressive validation accuracy, PNT/D requires the most data, followed by PNT/U, QC/U, and QC/D in order for the hybrid to dominant the baseline. In short, over the course of the whole annotation project, the hybrid pipeline performs better for automatic batch annotation (held-out accuracy) and interactive pre-annotation assistance (progressive validation accuracy) than does the baseline pipeline and dominates the baseline on average after 700 tokens or less have been annotated.

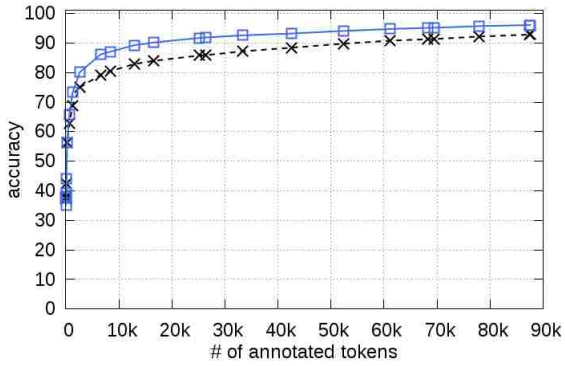
	PNT/D	PNT/U	QC/D	QC/U
Held-out				
snapshot RO+SEQ	658; 512 $\pm$ 192	231; 189 $\pm$ 73	* 4; 42 $\pm$ 21	169; 127 $\pm$ 165
complete RO+SEQ	* 22; 410 $\pm$ 225	* 22; 166 $\pm$ 74	* 4; 36 $\pm$ 19	† 169; 134 $\pm$ 160
RO+TOK	500; 440 $\pm$ 120	200; 160 $\pm$ 126	5; 26 $\pm$ 20	200; 131 $\pm$ 131
Progressive Validation				
RO+SEQ	*† 289; 656 $\pm$ 251	237; 210 $\pm$ 111	* 34; 124 $\pm$ 55	† 140; 155 $\pm$ 81
RO+TOK	*† 294; 684 $\pm$ 270	† 260; 211 $\pm$ 122	† 154; 141 $\pm$ 67	† 180; 175 $\pm$ 83

Table 7.7: Token after which overall and unknown accuracy of the hybrid pipeline is always greater than that of the baseline pipeline in the reading order experiment scenarios. Average hybrid dominance point values are formatted as <*natural*>; <*mean of 10 random*> $\pm$ <*std. dev. of 10 random*>. Asterisks (*) indicate significant difference between natural reading order and random reading order using a two-tailed t-test with $\alpha = 0.01$. Daggers (†) indicate scenarios in which the unknown accuracy statistics for the 10 random repetitions are slightly different than from the reported overall accuracy values.

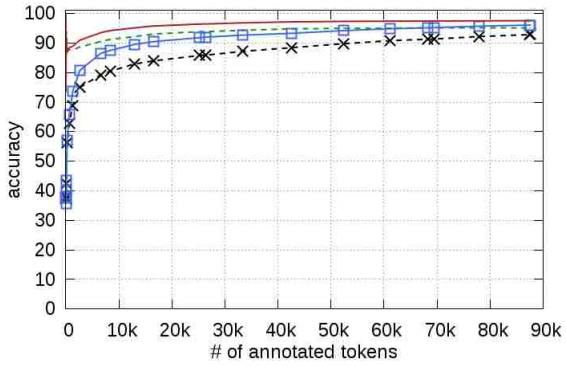
The hybrid pipeline achieves consistently higher *average* predictive accuracy than the baseline pipeline. We compute the average predictive accuracy for a pipeline as the ratio of the area under the test-share-scaled accuracy curve to the total area under the test share curve. Table 7.8 reports the average overall and unknown held-out and progressive validation accuracy for each pipeline in each data set and each reading order experiment scenario.[4] Average accuracy values are reported out to a hundredth of a percent

[4]The average known unambiguous and known ambiguous accuracy values are not reported in Table 7.8 since the measures are about the same for the two pipelines.

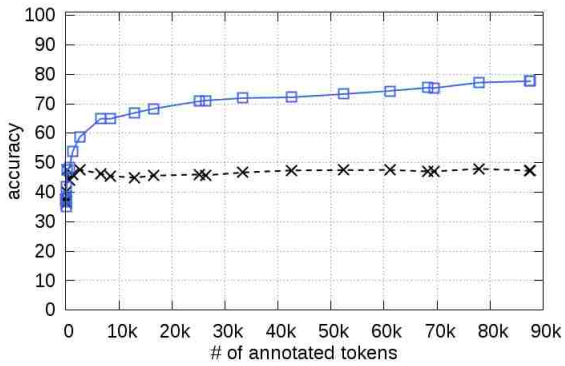
(a) Snapshot RO+SEQ Overall



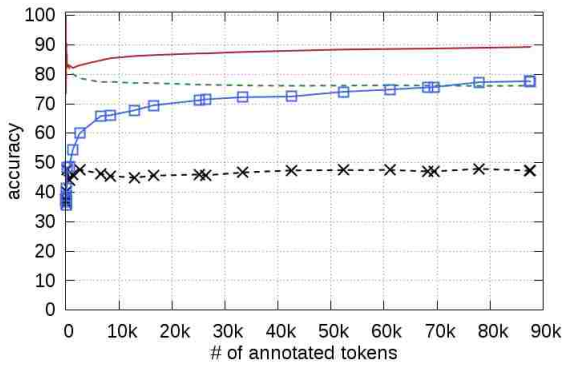
(b) Complete RO+SEQ Overall



(c) Snapshot RO+SEQ Unknown



(d) Complete RO+SEQ Unknown



(e) Legend

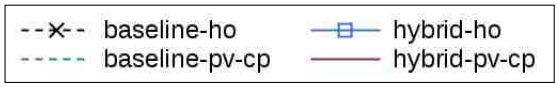


Figure 7.9: Overall and unknown accuracy learning curves for the PNT/D data set using the snapshot RO+SEQ scenario (evaluation configuration 1) and complete RO+SEQ scenario (configuration 2). The curves are representative of the trends for all data sets and for the RO+TOK scenario (configuration 3). Legend abbreviations: *"ho"* = held-out; *"pv"* = progressive validation; *"cp"* = correction propagation.

so that the standard deviation in overall accuracy is not unduly inflated because of rounding, although the practical difference at a hundredth of a percent is only a few word tokens or types. Across all data sets, all reading order experiment scenarios, and both accuracy measures, the hybrid pipeline achieves an average overall accuracy 2.52 to 8.70 percentage points higher than the baseline and an average unknown accuracy 9.01 to 31.85 percentage points higher than the baseline. The respective performance of the baseline and hybrid pipelines in all experiment scenarios is roughly the same, although for the hybrid the held-out average accuracy is regularly higher when trained incrementally after each sequence or token (H2 and H3 rows) and the progressive validation average accuracy is higher when trained incrementally after each token. Model performance using the natural reading order is significantly different than using a random reading order for both pipelines and for both overall accuracy measures (two-tailed t-tests and $\alpha = 0.01$). Model performance on unknown word types using the natural reading order is significantly different than using a random reading order for the hybrid pipeline in terms of held-out average accuracy, and usually for both pipelines in terms of progressive validation average accuracy. For the hybrid pipeline, a random reading order yields higher held-out average accuracy by 0.4–1.5 percentage points whereas the natural reading order usually yields higher progressive validation average accuracy by 0.1–2 percentage points. Thus, given the assumption that an annotation project will be conducted using some reading order, it would be better to use a random reading order if the goal is to cost-effectively build a model which can automatically generalize without further human input (held-out accuracy), whereas if the primary goal is to manually annotate and/or validate a specific corpus then the natural reading order would be better (progressive validation accuracy).

The incremental training algorithm for the MEMM model is suitable to support interactive CDL pre-annotation. Figure 7.10 shows the per-update training time learning curves for the snapshot RO+SEQ and complete RO+SEQ simulations for the PNT/D data sets. The training time learning curve for the RO+TOK simulation (not shown) is similar to the RO+SEQ simulation in overall trend and up-down behavior but with many more model updates. The baseline pipeline can be trained from scratch at interactive speeds since the maximum batch training time is around one second, but using the incremental training strategy the per-update time is even shorter at about a hundredth of a second. As shown in Figure 7.10a, the time to train the hybrid pipeline from scratch is not interactive, with training times exceeding our desired 10 second maximum mark after 10,000 annotated tokens, and exceeding 100 seconds after 50,000 annotated tokens. As shown in Figure 7.10b, using the incremental training algorithm, and updating after each sequence is annotated, the per-update incremental training time is below 10 seconds for the whole annotation project.

Held-out

	PNT/D	PNT/U	QC/D	QC/U
Overall				
B1,2	* 87.20; 88.63 ± 0.06	* 86.99; 88.23 ± 0.10	* 75.37; 76.68 ± 0.36	* 76.27; 76.89 ± 0.31
B3	* 87.19; 88.64 ± 0.06	* 86.96; 88.23 ± 0.11	* 75.53; 76.73 ± 0.35	* 76.39; 76.93 ± 0.29
H1	* **92.00**; **93.11** ± 0.08	* **90.84**; **91.98** ± 0.05	* **84.00**; **85.00** ± 0.11	* **82.03**; **82.67** ± 0.08
H2	* **92.13**; **93.21** ± 0.08	* **91.02**; **92.09** ± 0.05	* **84.14**; **85.15** ± 0.11	* **82.21**; **82.96** ± 0.07
H3	* **92.10**; **93.20** ± 0.07	* **91.02**; **92.09** ± 0.04	* **84.24**; **85.18** ± 0.10	* **82.25**; **82.99** ± 0.06
H>B	4.88; 4.54	3.98; 3.82	8.70; 8.41	5.85; 5.87
Unknown				
B1,2	46.52; 46.90 ± 0.35	45.90; 45.72 ± 0.39	27.03; 27.72 ± 1.22	34.11; 33.31 ± 1.26
B3	* 46.50; 46.96 ± 0.32	45.83; 45.79 ± 0.47	27.40; 27.80 ± 1.17	34.51; 33.37 ± 1.19
H1	* **68.92**; **70.60** ± 0.37	* **65.77**; **67.73** ± 0.34	* **58.17**; **59.20** ± 0.28	* **60.07**; **60.92** ± 0.26
H2	* **69.52**; **71.11** ± 0.34	* **66.69**; **68.41** ± 0.29	* **58.71**; **59.76** ± 0.26	* **60.86**; **62.25** ± 0.21
H3	* **69.43**; **71.07** ± 0.34	* **66.76**; **68.41** ± 0.29	* **58.96**; **59.82** ± 0.27	* **60.98**; **62.30** ± 0.22
H>B	22.78; 24.01	20.53; 22.44	31.46; 31.85	26.39; 28.49

Progressive Validation

	PNT/D	PNT/U	QC/D	QC/U
Overall				
B2	* 93.90; 93.42 ± 0.16	* 93.78; 93.11 ± 0.09	* 87.52; 86.24 ± 0.23	* 89.22; 88.07 ± 0.30
B3	* 94.01; 93.55 ± 0.15	* 93.90; 93.26 ± 0.09	* 87.90; 86.52 ± 0.22	* 89.49; 88.29 ± 0.29
H2	* **96.39**; **95.95** ± 0.04	* **96.31**; **95.70** ± 0.04	* **92.28**; **91.83** ± 0.05	* **92.56**; **92.16** ± 0.07
H3	* **96.50**; **96.05** ± 0.04	* **96.40**; **95.80** ± 0.04	* **92.49**; **92.03** ± 0.05	* **92.72**; **92.31** ± 0.05
H>B	2.49; 2.52	2.52; 2.57	4.68; 5.55	3.29; 4.06
Unknown				
B2	* 76.88; 77.87 ± 0.53	* 75.66; 76.13 ± 0.38	* 67.96; 66.54 ± 0.57	* 72.45; 70.53 ± 0.85
B3	* 76.94; 77.97 ± 0.55	* 75.76; 76.24 ± 0.40	* 68.37; 66.80 ± 0.58	* 72.61; 70.69 ± 0.85
H2	**86.96**; **86.86** ± 0.13	* **86.77**; **86.07** ± 0.15	* **81.62**; **81.46** ± 0.10	* **83.96**; **83.68** ± 0.14
H3	* **87.18**; **87.00** ± 0.15	* **86.91**; **86.19** ± 0.14	* **81.85**; **81.73** ± 0.11	* **84.17**; **83.86** ± 0.11
H>B	10.16; 9.01	11.13; 9.95	13.57; 14.93	11.54; 13.16

Table 7.8: Held-out and progressive validation average accuracy for the baseline (B) and hybrid (H) models in the reading order experiment scenarios: evaluation configuration 1 = snapshot RO+SEQ, configuration 2 = complete RO+SEQ, and configuration 3 = RO+TOK. Average accuracy values are formatted as <*natural*>; <*mean of 10 random*> ± <*std. dev. of 10 random*>. "H>B" rows provide the average percentage point difference between the hybrid and baseline pipeline average accuracy values. Bold entries indicate the best performing pipeline and experiment scenario. Asterisks (*) indicate significant difference between natural reading order and random reading order using a two-tailed t-test with $\alpha = 0.01$.

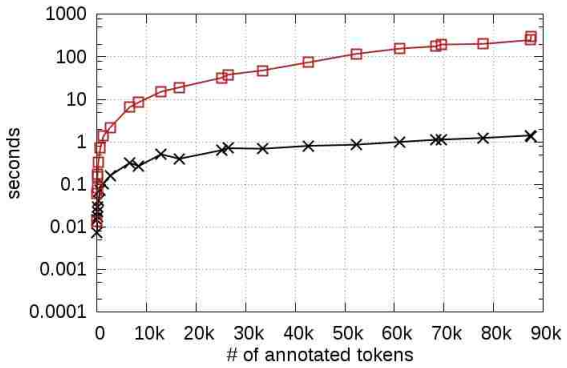
The training time learning curves for the other data sets are similar, except that for the QC/D data set the incremental training times do start the exceed the 10 second mark towards the end of the annotation project. Table 7.9 reports the average time to train each pipeline, computed as the average area under the curve where the x-axis is measured by model updates in each respective simulation instead of being measured by the number of annotated tokens. The average training time values demonstrate the same observations as the learning curves: (1) training the baseline is easy, with average per-update times less than one second; (2) training the hybrid from scratch per-update would not be suitable for interactivity, with average training times ranging from close to a minute (56 seconds) for QC/U to close to five-and-a-half minutes (330 seconds) for QC/D; and (3) training the hybrid incrementally is suitable for interactivity if trained in parallel to providing pre-annotations since average training times are under 10 seconds for all data sets. Furthermore, training the hybrid model after each token is annotated instead of after each sequence brings the average training time per-update down by roughly a factor of 6, but since there are 12–13 tokens per verse in our data sets the total time to simulate with the token change model update strategy takes about twice as long as simulating with the sequence change model update strategy as seen in Table 7.10. Although most of the differences between the natural reading order and random reading order training times are statistically significant (two-tailed t-test with $\alpha = 0.01$), the two reading orders are not practically different in terms of training times.

	PNT/D	PNT/U	QC/D	QC/U
B1	* 0.80; 0.82 ± 0.01	* 0.77; 0.82 ± 0.01	* 0.72; 0.65 ± 0.01	* 0.70; 0.61 ± 0.01
B2	* 0.01; 0.01 ± 0.00	* 0.01; 0.01 ± 0.00	* 0.01; 0.01 ± 0.00	* 0.01; 0.01 ± 0.00
B3	* 0.01; 0.01 ± 0.00	* 0.01; 0.01 ± 0.00	* 0.01; 0.01 ± 0.00	* 0.01; 0.01 ± 0.00
H1	* 97.56; 112.19 ± 2.90	* 92.09; 104.16 ± 2.31	* 336.16; 266.70 ± 12.38	* 70.28; 56.65 ± 1.30
H2	* 1.17; 1.24 ± 0.02	* 0.68; 0.69 ± 0.01	* 3.89; 3.70 ± 0.03	* 0.58; 0.57 ± 0.00
H3	* 0.18; 0.18 ± 0.00	* 0.10; 0.10 ± 0.00	0.62; 0.62 ± 0.00	* 0.10; 0.10 ± 0.00

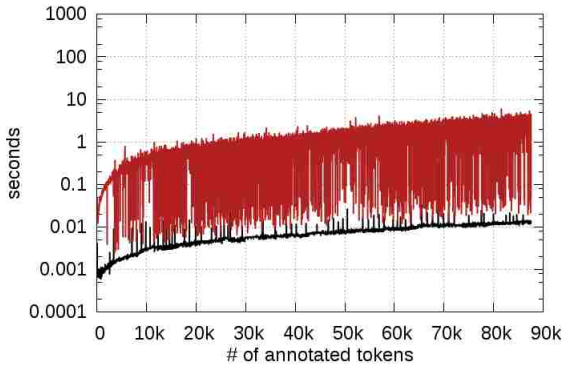
Table 7.9: Average training time in seconds per model update for the baseline (B) and hybrid (H) models in the reading order experiment scenarios: evaluation configuration 1 = snapshot RO+SEQ, configuration 2 = complete RO+SEQ, configuration 3 = RO+TOK. Average training time values are formatted as <*natural*>; <*mean of 10 random*> ± <*std. dev. of 10 random*>. The baseline is expected to always have a shorter training time than the hybrid model. Asterisks (*) indicate significant difference between natural reading order and random reading order using a two-tailed t-test with $\alpha = 0.01$.

In summary, this section has demonstrated that using the hybrid pipeline yields better accuracy than the baseline and that incrementally training the hybrid makes it viable to use for interactive pre-annotation

(a) Snapshot RO+SEQ Batch Time



(b) Complete RO+SEQ Incremental Time



(c) Legend

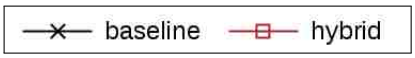


Figure 7.10: Training time learning curves for the PNT/D data set using the snapshot RO+SEQ scenario (evaluation configuration 1) and the complete RO+SEQ scenario (configuration 2). The curves are representative of the batch and incremental training time trends for all data sets and for the RO+TOK scenario (configuration 3). Note that Figure 7.10b only uses the curve colors from the legend.

	PNT/D		PNT/U		QC/D		QC/U	
B2	2:52;	3:02	2:60;	2:54	2:29;	2:24	2:16;	2:12
B3	13:17;	14:58	11:50;	13:14	10:04;	10:33	8:16;	8:41
H2	2:35:11;	2:43:22	1:33:37;	1:34:30	6:31:26;	6:14:48	1:07:29;	1:06:54
H3	5:19:52;	5:23:19	3:01:14;	3:06:05	13:10:01;	13:15:06	2:23:43;	2:25:19

Table 7.10: Total simulation time for the baseline (B) and hybrid (H) models in the reading order experiment scenarios: evaluation configuration 2 = complete RO+SEQ, configuration 3 = RO+TOK. Total simulation time values are reported as *<natural>; <mean of 10 random>* in *hours:minutes:seconds* format. Variation and statistical significance are elided because of space and because there is no practical difference in the total simulation times.

assistance. The goals of the annotation project should inform the decision of whether to use the natural reading order or a random reading order: a random reading order is good for quickly obtaining a model that generalizes well and is a baseline approach for active learning data selection strategies (e.g., Haertel, 2013), whereas the natural reading order is better in the case that human annotators will validate every token in the corpus.

7.5.3 Word Frequency Selection Strategy Experiments

In this section we compare incremental training using the reading order data selection strategy to descending and ascending word type frequency selection strategies. The experiment scenarios are all the evaluation configuration 3 scenarios: reading order and token change (RO+TOK), descending word frequency and token change (DWF+TOK), descending word frequency and word type change (DWF+WTC), ascending word frequency and token change (AWF+TOK), and ascending word frequency and word type change (AWF+WTC). We envision that the word frequency data selection strategies would be used with a keyword-in-context (KWIC) search tool (see Figure 1.1b). The bulk of our analysis looks at the token change experiments which correspond to a human annotating one token at a time which allows the pre-annotation suggestions for later selected tokens to be updated in light of what the human as already annotated. We also briefly look at the word type change experiments to determine how much model performance is affected if the whole set of items in the KWIC list were annotated at one time such that the model could not provide updated pre-annotations while the human worked through the list. The two extremes—annotating each token versus annotating the whole set of tokens at once—provide bounds on the more likely preferred work flow of selecting a set of tokens to receive the same annotation and then selecting and annotating another set of tokens and so on. In all scenarios the hybrid pipeline's unknown component models use *first input-label pairs* filter and *max 10* stopping criteria. In the word frequency scenarios the word tokens of each type have to be ordered in some way. We compared ordering the tokens by the natural reading order and random reading orders for the DWF+TOK scenario and determined that there is not much statistical difference between the two. Thus, we conduct the analysis for only the natural reading order.

There are four principal conclusions from the analysis. (1) Word frequency data selection strategies are best used to support annotation projects where humans will validate each annotation. (2) The DWF+TOK scenario usually yields the best overall and unknown accuracy values for both the baseline and hybrid pipelines, with the hybrid performing better than the baseline. (3) Human annotators should be

encouraged to commit annotations as frequently as possible. (4) The hybrid pipeline can be trained interactively using any of the scenarios but the shortest average training time is with the DWF+TOK or RO+TOK scenarios.

The partition of the held-out and progressive validation test data into different ambiguity cases have very different trends. Figure 7.11 shows the test share trends for the DWF+TOK scenario for the four data sets, and Figure 7.12 shows the test share trends for the AWF+TOK scenario for the four data sets. The DWF+WTC and AWF+WTC scenarios have similar held-out trends to the DWF+TOK and AWF+TOK scenarios respectively, but the progressive validation trends are drastically different since the scenarios test the extreme case where all of the word tokens are unknown when selected for annotation thus the unknown share is 100% through the whole simulation. The held-out test share trends are fairly linear compared to the logarithmic-looking test share trends for RO+TOK (see Figure 7.7), but in all cases the unknown shares start out the largest and gradually decrease over time, and the known shares start out the smallest and gradually increase over time. The progressive validation test share trends for AWF+TOK are similar to the held-out trends except that the unknown shares make up a majority of the data for longer at the outset of the project when the singleton word types are annotated, and then thereafter the unknown shares follow a logarithmic-like decay and the known unambiguous shares follow a logarithmic-like growth, similar to the RO+TOK trends. The progressive validation test share trends for the DWF+TOK scenario are very different. The unknown share is very small for the first half of the annotation project and then picks up when the singleton word types are annotated. In the diacritized data sets the known unambiguous shares are the largest throughout the whole annotation project, although in the QC/D data set the known unambiguous shares dip as the known ambiguous shares rise at the beginning of the project. In the undiacritized data sets the known ambiguous shares are the largest early on and then the known unambiguous shares gain dominance after 5,000–20,000 annotated tokens. The fact that the known ambiguous shares are large or are the largest early on in the PNT/U, QC/D, and QC/U data sets indicates that the most frequent word types which are annotated first also happen to be quite ambiguous; granted, some of the top most frequent word types are not ambiguous which is why the known ambiguous trend lines, especially for QC/D and QC/U, have dips and peaks in the first 10,000 annotated tokens before monitonically decreasing in percentage share of the annotated corpus. The differences in the held-out and progressive validation test share trends demonstrate two things: (1) that the simulations were coded correctly since the trends have intuitive explanations, and (2) that in order to effectively compare accuracy metrics broken down by ambiguity case, that the accuracy

needs to be scaled by the test share, which is what we do in calculating average accuracy via a ratio of area under the curve.

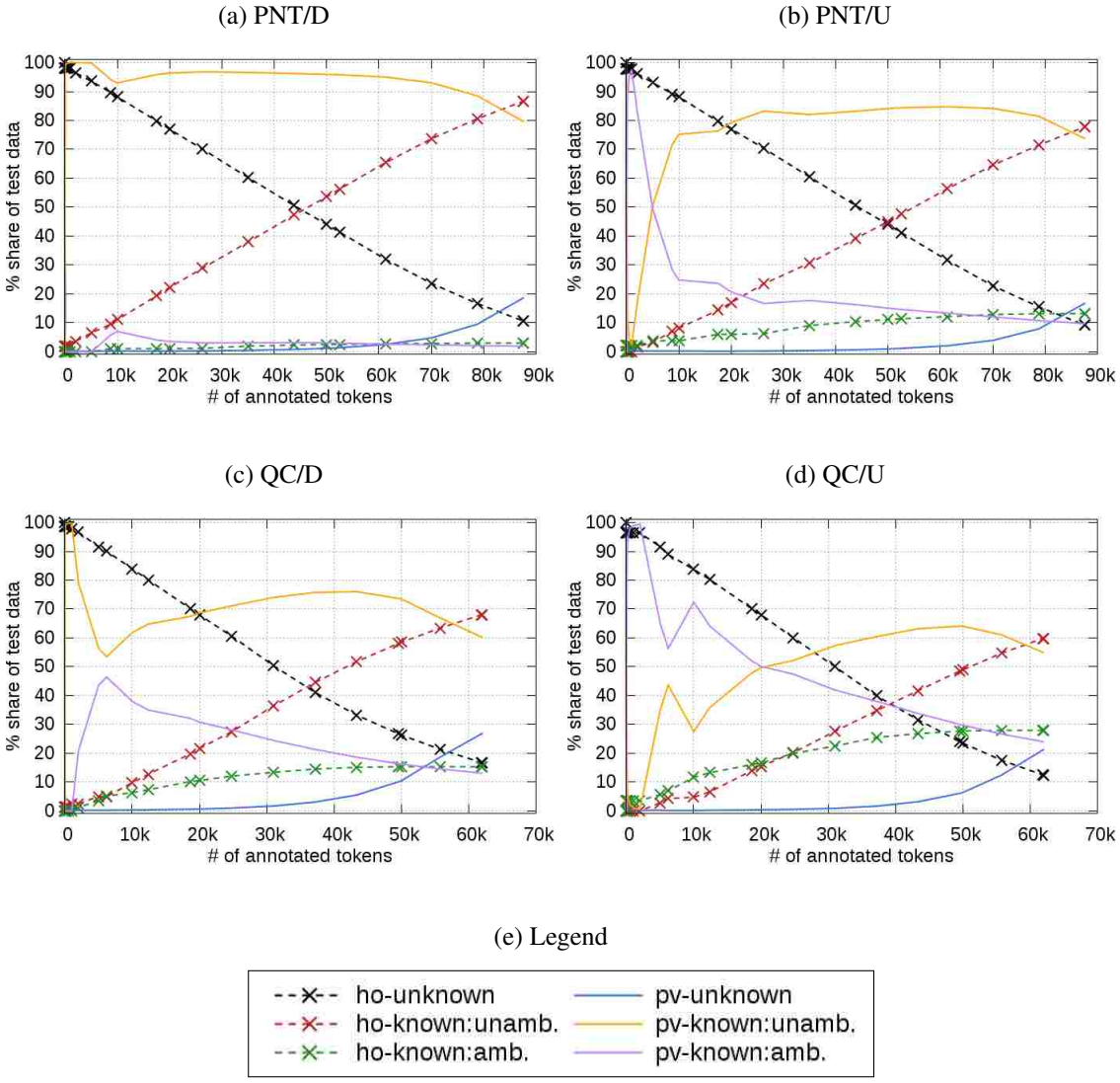
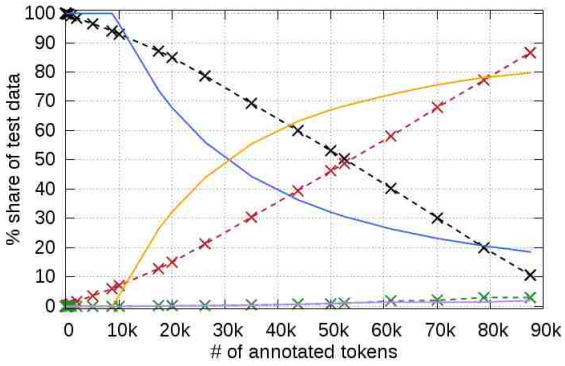


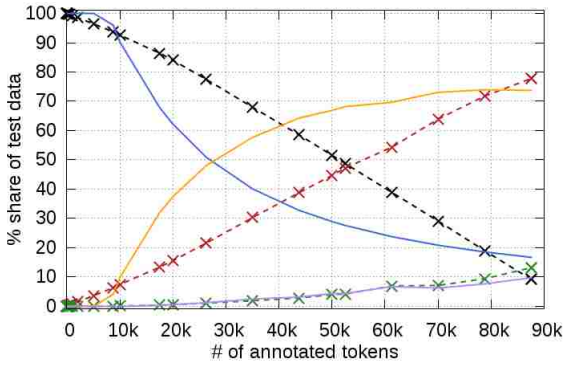
Figure 7.11: Percentage share of held out and progressive validation test data per unknown, known unambiguous, and known ambiguous category in the DWF+TOK scenario for each of the four data sets. The held-out trends for DWF+WTC are similar, and the progressive validation unknown trend is 100% for the whole annotation project.

The accuracy learning curves for the AWF+TOK scenario are similar to the RO+TOK scenario, but the learning curves for the DWF+TOK scenario are very different. Figure 7.13 shows the overall and unknown accuracy learning curves for the PNT/D data set using the DWF+TOK and AWF+TOK scenarios. Table 7.11 reports the token after which the hybrid pipeline's accuracy is greater than the baseline pipeline's

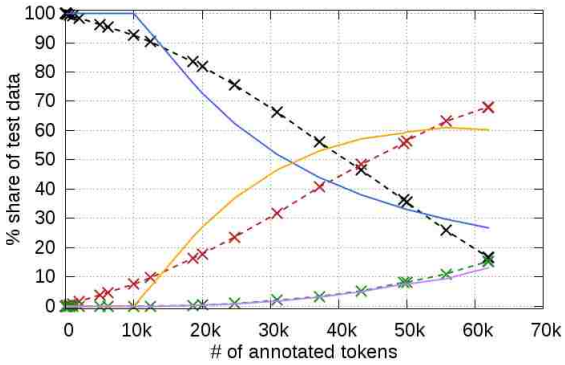
(a) PNT/D



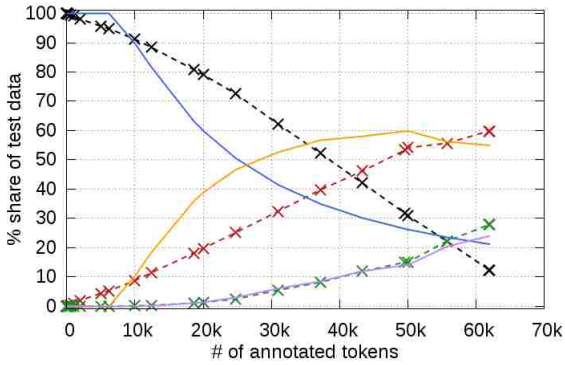
(b) PNT/U



(c) QC/D



(d) QC/U



(e) Legend

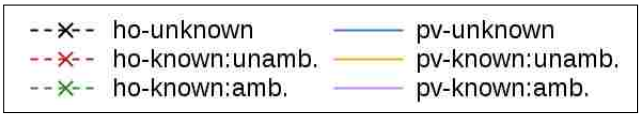


Figure 7.12: Percentage share of held out and progressive validation test data per unknown, known unambiguous, and known ambiguous category in the AWF+TOK scenario for each of the four data sets. The held-out trends for AWF+WTC are similar, and the progressive validation unknown trend is 100% for the whole annotation project.

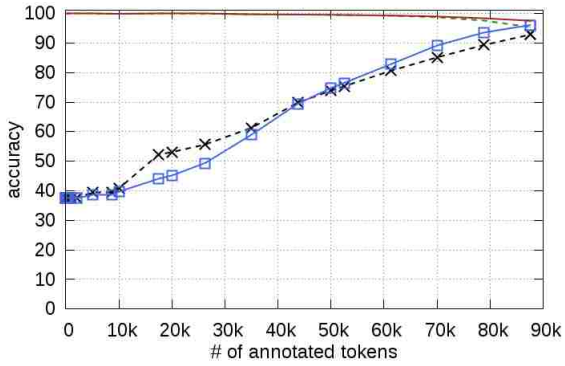
accuracy for the remainder of the annotation project. The accuracy trends are similar for the DWF+WTC and AWF+WTC scenarios and for the four data sets. The learning curves for the AWF+TOK scenario have a similar shape to the RO+TOK learning curves (see Figure 7.9) in that the hybrid does better than the baseline by both accuracy metrics and the progressive validation accuracy is greater than the counterpart held-out accuracy. Measured by held-out accuracy, the hybrid in the AWF+TOK scenario performs better than the baseline with 2,000 annotated tokens or less. Measured by progressive validation accuracy, the hybrid in the AWF+TOK scenario performs better than the baseline almost immediately after about 100 tokens have been annotated (see Table 7.11). For the DWF+TOK scenario, the held-out and progressive validation learning curves have very different shapes: the progressive validation curves start out close to 100% and remain high whereas the held-out trends start low and increase slowly. Furthermore, in the DWF+TOK scenarios the hybrid performs about the same or worse than the baseline for more than half of the annotation project. This difference in accuracy trends demonstrates that the descending word frequency data selection strategy is not a cost-effective strategy for acquiring annotated data to train a model which generalizes well as compared to the reading order strategy (held-out accuracy), but rather the descending word frequency strategy is designed to aid human annotators to quickly create or validate the annotations in a large portion of a corpus (progressive validation accuracy).

	PNT/D	PNT/U	QC/D	QC/U
Held-out				
RO+TOK	500	200	5	200
DWF+TOK	50,000	43,775	18,585	30,975
DWF+WTC	48,864	41,108	14,137	28,773
AWF+TOK	2,000	1,000	1	100
AWF+WTC	2,000	2,000	1	100
Progressive Validation				
RO+TOK	294	260	154	180
DWF+TOK†	50,871	45,640	21,905	23,811
DWF+WTC	62,307	42,680	21,770	13,170
AWF+TOK	90	86	6	95
AWF+WTC	90	86	6	95

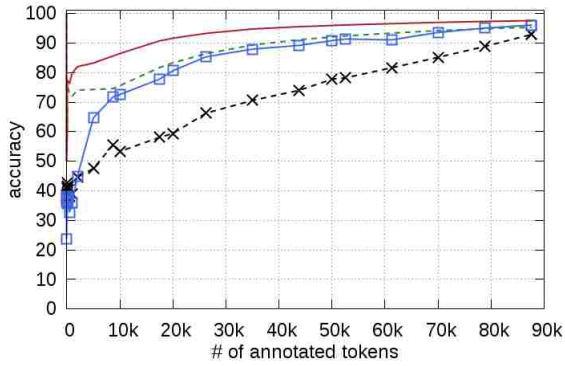
Table 7.11: Token after which overall and unknown accuracy of the hybrid pipeline is always greater than that of the baseline pipeline in the evaluation configuration 3 experiment scenarios. †The unknown accuracy hybrid dominance points are smaller (earlier) than the overall accuracy points reported in the table: 49,050 for PNT/D, 38,742 for PNT/U, 19,623 for QC/D, and 13,028 for QC/U.

Measured by progressive validation accuracy, the DWF+TOK scenario yields the highest overall and

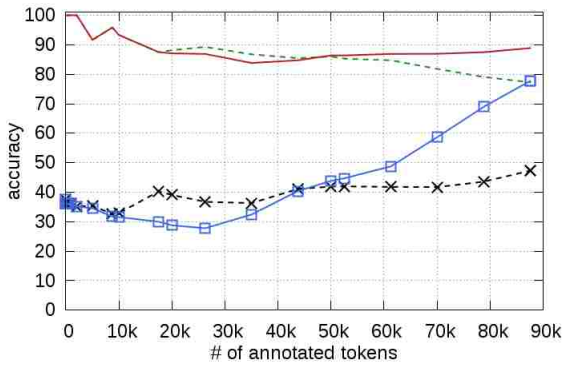
(a) DWF+TOK Overall



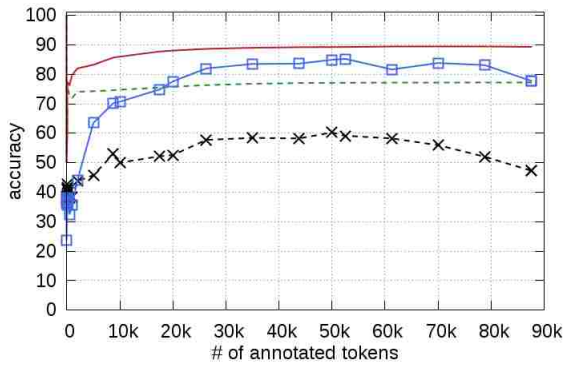
(b) AWF+TOK Overall



(c) DWF+TOK Unknown



(d) AWF+TOK Unknown



(e) Legend

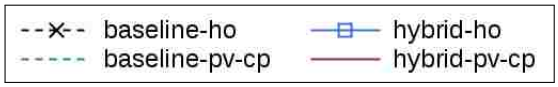


Figure 7.13: Overall and unknown accuracy learning curves for the PNT/D data set using the DWF+TOK and AWF+TOK simulation scenarios. The curves are representative of the trends for all data sets and for the DWF+WTC and AWF+WTC scenarios. Legend abbreviations: *"ho"* = held-out; *"pv"* = progressive validation; *"cp"* = correction propagation.

unknown average accuracy values for both the baseline and hybrid pipelines. Table 7.12 reports the average accuracy for each pipeline on each data set for each experiment scenario. The word frequency data selection strategies are not intended to optimize overall held-out accuracy as evident in the fact that the reading order scenario has the highest overall held-out average accuracy. Interestingly though, the ascending word frequency data selection strategy usually achieves the best unknown held-out accuracy, perhaps because the whole set of diverse singleton word types is annotated first in the annotation project such that the model has greater ability to generalize to novel word types in a regularized manner. In terms of overall progressive validation accuracy the DWF+TOK scenario yields the highest average accuracy for both the baseline and the hybrid on all of the data sets. The DWF+TOK scenario also yields the highest unknown progressive validation accuracy for the hybrid pipeline and for the baseline pipeline on the PNT data sets. The unknown progressive validation accuracy for the word type change scenarios (DWF+WTC and AWF+WTC) is the same as the overall accuracy but is not comparable to the token change scenarios inasmuch as all of the data is considered as unknown instead of only being a variable portion of all of the data. While the hybrid overall accuracy is greater than the baseline accuracy using the DWF+TOK scenario, the difference between the pipelines' average accuracy values is less than one percentage point so the hybrid provides correct annotations for only about 10 to 100 tokens more than does the baseline. If human annotators intend to strictly use the descending word frequency data selection strategy then the baseline pre-annotation assistance may be sufficient and is simpler to implement, but since human annotators will likely deviate and use a combination of descending word frequency and reading order data selection strategies then it is would be beneficial to maintain and utilize the hybrid pipeline which generalizes better than the baseline.

In order to have better pre-annotation assistance, human annotators should be encouraged to commit annotations as frequently as possible. This recommendation comes from comparing the word type change experiments to the token change experiments. Updating the model when the word type changes, which simulates the human annotating the whole set of tokens at one time instead of annotating each token one at a time, reduces average accuracy: the hybrid pipeline in the AWF scenarios is the most resilient, losing only 2-3 percentage points, whereas the baseline model in the AWF scenarios and both models in the DWF scenarios lose 8-20 percentage points. Thus, to have better pre-annotation support when using a KWIC-style annotation tool, the human annotators should be encouraged to commit annotations as frequently as possible. For example, say the annotator has mentally decided on an annotation for a token: if the annotator will commit that annotation before looking for other tokens that should be annotated the same way, then the

model can be retrained and provide updated pre-annotations for the other tokens for the whole list which will help the human identify the other tokens which should be (or should not be) annotated the same way.

	Held-out				Progressive Validation			
	PNT/D	PNT/U	QC/D	QC/U	PNT/D	PNT/U	QC/D	QC/U
Baseline Overall								
RO+TOK	**87.19**	**86.96**	**75.53**	**76.39**	94.01	93.90	87.90	80.49
DWF+TOK	67.42	67.52	51.76	57.44	**99.19**	**98.77**	**96.42**	**94.70**
DWF+WTC	67.55	67.36	51.52	57.37	88.82	89.13	74.84	82.53
AWF+TOK	72.27	68.70	53.45	50.06	88.24	88.17	79.34	83.10
AWF+WTC	72.29	68.63	53.75	50.66	79.98	80.20	68.79	70.22
Hybrid Overall								
RO+TOK	**92.10**	**91.02**	**84.24**	**82.25**	96.50	96.40	92.49	92.72
DWF+TOK	66.96	66.83	56.40	60.21	**99.40**	**98.98**	**97.24**	**95.18**
DWF+WTC	66.91	66.69	56.09	60.10	88.85	89.99	78.57	86.90
AWF+TOK	85.22	82.46	77.23	74.14	93.58	93.05	89.20	90.11
AWF+WTC	85.11	81.57	77.03	74.94	91.40	90.84	86.68	87.19
Baseline Unknown								
RO+TOK	46.50	45.83	27.40	**34.15**	76.94	75.76	**68.37**	**72.61**
DWF+TOK	38.18	39.05	14.58	29.16	**80.39**	**78.92**	67.83	71.10
DWF+WTC	38.54	38.93	14.23	29.23	NC	NC	NC	NC
AWF+TOK	**54.18**	**48.50**	29.30	24.02	75.72	74.62	66.24	70.50
AWF+WTC	53.97	47.86	**29.57**	24.87	NC	NC	NC	NC
Hybrid Unknown								
RO+TOK	69.43	66.76	58.96	60.98	87.18	86.91	81.85	84.17
DWF+TOK	37.29	37.71	23.30	34.53	**87.72**	**87.72**	**83.32**	**85.10**
DWF+WTC	37.31	37.63	22.82	34.47	NC	NC	NC	NC
AWF+TOK	**76.33**	**72.47**	**66.75**	63.97	87.13	85.82	82.86	84.18
AWF+WTC	75.93	70.47	66.39	**65.33**	NC	NC	NC	NC

Table 7.12: Held-out and progressive validation average accuracy for the baseline (B) and hybrid (H) models in the evaluation configuration 3 experiments where the pipeline is trained with partially annotated sequences. "NC" entries indicate that the value from the word type change (WTC) experiment is not comparable to the token change (TOK) experiments because the unknown accuracy is the same as the overall accuracy. Bold entries indicate the best performing scenario for each pipeline type; the hybrid average accuracy is consistently greater than the baseline average accuracy.

The incremental training algorithm for the hybrid model is suitable to support interactive pre-annotation in any experiment scenario, but the DWF+TOK and RO+TOK scenarios have the shortest training time. Figure 7.14 shows the per-update training time learning curves for the four word frequency scenarios for the PNT/D data set. The training time trends for the other data sets are similar, except that for the QC/U data set the training time trend regularly modulates over the desired 10 second maximum mark for accom-

modating training in parallel to providing pre-annotations. The baseline pipeline can be trained at interactive speeds with each model update taking around a tenth of a second or less. The hybrid pipeline can also be trained at interactive speeds in the token change scenarios (DWF+TOK and AWF+TOK) since the training time is less than 10 seconds. The training times are suitably interactive for most of the annotation in the word type change scenarios (DWF+WTC and AWF+WTC) except that they dramatically and consistently rise above 10 seconds at the beginning of the DWF+WTC scenario and at the end of the AWF+WTC scenario. Another reason why the human annotators should be encouraged to commit annotations as frequently as possible is to get shorter training times as in the token change scenarios instead of longer training times as in the word type change scenarios. Table 7.13 reports the average time to train each pipeline, computed as the average area under the curve. The average training time values demonstrate the same observations as the learning curves: (1) training the baseline is easy, with average per-update times less than one second; (2) training the hybrid in each scenario is suitable for interactivity when training and pre-annotation prediction are done in parallel with average training time being under 10 seconds for all data sets; (3) yet the DWF+TOK and RO+TOK scenarios have the shortest per-update training times. Furthermore, in terms of total simulation time which is also reported in Table 7.13, it is faster to simulate the hybrid model in the reading order and descending word frequency scenarios (RO+TOK, DWF+TOK, and DWF+WTC) than the ascending word frequency scenarios (AWF+TOK and AWF+WTC).

In summary, this section has demonstrated that annotating a corpus using a word frequency data selection strategy can be done with high quality interactive pre-annotation assistance. The pre-annotation cumulative accuracy (progressive validation) is most accurate when using the descending word frequency data selection strategy, even more so than following the natural reading order, provided the human annotators commit annotations as frequently as possible.

7.6 Conclusion

In this chapter we demonstrated that context-sensitive, high quality pre-annotation assistance is interactively viable by training the hybrid MEMM pipeline incrementally. We evaluated model performance in terms of accuracy and training time by simulating corpus annotation of the diacritized and undiacritized Peshitta New Testament and Quranic Corpus data sets. These data sets are representative of low-resource language tasks. In all the cases we tried, the hybrid pipeline achieves higher overall average accuracy than the base-

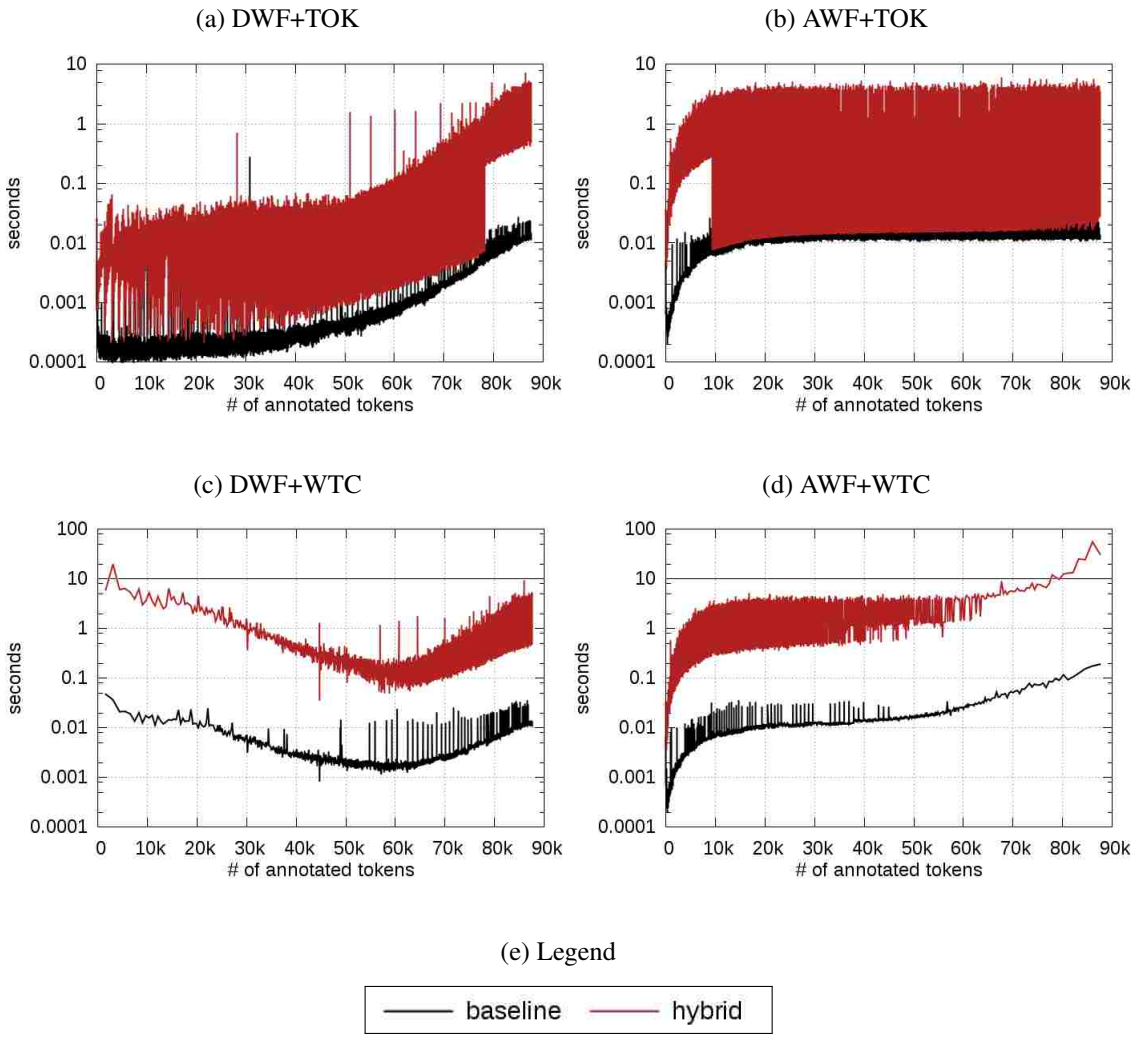


Figure 7.14: Training time learning curves for the PNT/D data set using the word frequency simulation scenarios. The curves are representative of the trends for all data sets.

	Average Training Time				Total Simulation Time			
	PNT/D	PNT/U	QC/D	QC/U	PNT/D	PNT/U	QC/D	QC/U
Baseline								
RO+TOK	0.007	0.006	0.008	0.006	13:17	11:50	10:04	8:16
DWF+TOK	**0.002**	**0.001**	**0.002**	**0.001**	4:42	4:05	4:06	3:16
DWF+WTC	0.007	0.005	0.006	0.005	4:09	3:38	3:48	2:59
AWF+TOK	0.011	0.010	0.011	0.009	19:22	16:55	13:47	11:19
AWF+WTC	0.006	0.006	0.007	0.005	4:14	3:44	3:59	3:12
Hybrid								
RO+TOK	**0.176**	**0.095**	**0.623**	**0.099**	5:19:52	3:01:14	13:10:01	2:23:43
DWF+TOK	**0.178**	**0.099**	**0.666**	**0.105**	5:09:20	3:00:26	12:47:06	2:21:35
DWF+WTC	0.946	0.582	2.462	0.483	5:20:09	3:06:39	13:13:17	2:26:14
AWF+TOK	0.375	0.212	1.366	0.196	13:01:24	6:51:08	29:17:10	4:40:33
AWF+WTC	0.986	0.601	2.684	0.444	8:46:17	4:22:08	19:03:06	3:04:49

Table 7.13: Average training time per model update (left) and total simulation time (right) for the baseline (top) and hybrid (bottom) models in the evaluation configuration 3 experiments where the model is trained with partially annotated sequences. Average training time is reported in seconds and total simulation time is reported in *hours:minutes:seconds* format. Bold entries indicate the shortest training time per pipeline type. The baseline is expected to always have a shorter training time than the hybrid model.

line pipeline. When using a reading order data selection strategy or the ascending word frequency strategy, the hybrid obtains and maintains dominance over the baseline in terms of held-out accuracy with 2,000 or fewer annotated tokens, and in terms of progressive-validation accuracy achieves dominance on average with less than 700 annotated tokens. When using the descending word frequency data selection strategy, the hybrid and baseline pipelines are comparable in their held-out and progressive validation accuracy performance sometimes until halfway through the annotation project, but the hybrid does eventually achieve higher accuracy. Using the natural reading order yields higher progressive validation accuracy than using a random reading order, but using the descending word frequency selection strategy yields even higher accuracy provided the human annotators commits decisions as frequently as possible. Each incremental hybrid model update usually takes less than 10 seconds, and so training the model in parallel to using an older copy of the model to predict pre-annotation should be perceived as interactive provided the time it takes the annotators to provide annotation validations and corrections is longer than 10 seconds. Therefore, human annotators would likely perceive pre-annotation assistance provided with the hybrid as being interactive and since the pre-annotations are high quality then the annotators should be able to annotate more quickly as demonstrated by Felt et al. (2013). Because the accuracy of the pre-annotation assistance improves quickly from initially zero annotated data, then annotators can immediately utilize interactive pre-annotation assis-

tance even with low-resource language tasks which have no existing annotated data; furthermore, since we evaluated on Syriac and Arabic our results are particularly promising for low-resource annotation tasks in those languages.

Chapter 8

Conclusion

The key contribution of the thesis is the development of interactive pre-annotation models for the corpus-dictionary linkage sequence tagging task. In this work we formulated the corpus-dictionary linkage (CDL) task with three sub-tasks executed in a pipeline: morphological segmentation, stem-to-lemma linkage, and homographic headword disambiguation. We first compared models for the stem-to-lemma linkage CDL sub-task, which is the most difficult of the three sub-tasks. We determined that the hybrid-Morfette model achieves the best accuracy and has the shortest batch training time of the compared models on the diacritized and undiacritized Peshitta New Testament and Quranic Corpus data sets. Despite the fact that the DirecTL+ model learns character-subsequence alignments which are more flexible than the edit scripts employed by hybrid-Morfette, DirecTL+ has lower or comparable accuracy and the batch training time in seconds takes an order of magnitude longer. Given the stem-to-lemma linkage results, we employed the same hybrid maximum entropy Markov model (MEMM) design for the other two CDL sub-tasks and combined the individual sub-task models into a pipeline. We developed an incremental training algorithm for MEMM model which simply uses any features available in each partially or fully annotated sequence in the training data: it does not do extra processing to interpolate missing tags and then learn from those interpolations. We evaluated the performance of the hybrid pipeline, trained with the incremental training algorithm, against the baseline pipeline on all four data sets and using a variety of data selection strategies for determining the order in which to annotate. We determined that the hybrid pipeline achieves higher progressive validation accuracy than the baseline almost immediately, with 92%–99% average accuracy depending on the data set and data selection strategy, and that each model update on average takes less than 10 seconds. Therefore, if a new model is trained in parallel to predicting pre-annotations with an old model, human users would likely perceive pre-annotation assistance provided with the hybrid as being interactive and the pre-annotations as being high quality which should allow them to annotate more quickly. Because the ac-

curacy of the hybrid increases quickly, even starting with zero annotated data, annotators can immediately utilize interactive pre-annotation assistance even with low-resource language tasks which have no existing annotated data. Given the manual data selection strategies that we simulated, the interactive pre-annotation assistance is most accurate on average if word types are annotated in order of descending frequency and if the annotators commit annotations as frequently as possible. The pre-annotation assistance is also very good when following the natural reading order or a random reading order, as could be done with simple active learning sample selection.

We finish by summarizing promising avenues to further improve interactive pre-annotation assistance for CDL. While the construction of an annotation system for CDL is outside the scope of this work, that is a desirable next step in the research path so as to validate that humans actually like interactive pre-annotation assistance and perceive it as being valuable. Additionally, an annotation system will facilitate other research, including: (1) supporting the next step of the Syriac Electronic Corpus project to link the writing of Ephrem the Syrian to the Jessie Payne Smith Compendious Syriac Dictionary; (2) collecting data with which to build sophisticated empirical models of annotator behavior in order to conduct better simulations; and (3) evaluating manual selection strategies and project management versus active learning. In the annotation system a ranked list of pre-annotations will be needed for each token. One possible way to rank the pre-annotations for a specific token using a sequence tagging model is to score each applicable label for the token by the annotation score for the whole sequence. The models should also be evaluated with some metric that considers a list of pre-annotations such as mean reciprocal rank or precision (accuracy) among the top n suggestions. Linear model hybrids, which have the potential to achieve better accuracy than the MEMM, could be developed for interactive use and be compared to the interactive MEMM hybrid. Using linear models would also facilitate analysis of regret bounds in a coactive learning scenario where human feedback is not optimal (Shivaswamy and Joachims, 2012; Sokolov et al., 2015). Model performance can be improved further with feature engineering. Most promising would be to develop pipeline component models which extract features from all previous input levels in the pipeline. Using the edit tree scripts from Chrupała (2008) would likely boost the accuracy of hybrid-Morfette. Furthermore, we are intrigued by the root-and-pattern features of dkrMorph (Lindgren, 2011) which are similar to the edit scripts already being used as tags in Morfette but which, if used as features, would enable the model to weight each candidate transformation relative to other applicable transformations. We welcome interested researchers to implement and investigate these topics in order to provide practical and quality interactive pre-annotation

assistance for corpus-dictionary linkage and other annotation tasks.

References

S. Abney. 1997. *Part-of-speech tagging and partial parsing*. In Corpus-Based Methods in Language and Speech Processing, volume 2, pp. 118–136. Springer Netherlands.

F. Ahmed and A. Nürnberger. 2011. *A web statistics based conflation approach to improve Arabic text retrieval*. In Proceedings of the Federated Conference on Computer Science and Information Systems, pp. 3–9. Institute of Electrical and Electronics Engineers.

V. Basile, J. Bos, K. Evang, and N. Venhuizen. 2012. *Developing a large semantically annotated corpus*. In Proceedings of the Eighth International Conference on Language Resources and Evaluation. European Language Resources Association, Istanbul, Turkey.

K. Black, E. K. Ringger, P. Felt, K. Seppi, K. Heal, and D. Lonsdale. 2014. *Evaluating lemmatization models for machine-assisted corpus-dictionary linkage*. In Proceedings of the Ninth International Conference on Language Resources and Evaluation. European Language Resources Association.

A. Blum, A. Kalai, and J. Langford. 1999. *Beating the hold-out: Bounds for k-fold and progressive cross-validation*. In Proceedings of the Twelfth Annual Conference on Computational Learning Theory, pp. 203–208. Association for Computing Machinery, New York, NY, USA.

K. Bontcheva, H. Cunningham, I. Roberts, A. Roberts, V. Tablan, N. Aswani, and G. Gorrell. 2013. *GATE teamware: A web-based, collaborative text annotation framework*. Language Resources and Evaluation, volume 47(4):pp. 1007–1029.

C. Brockelmann. 1895. *Lexicon Syriacum*. Reuther & Reichard.

T. Buckwalter. 2002. *Buckwalter Arabic morphological analyzer version 1.0 LDC2002L49*. Web Download. Philadelphia: Linguistic Data Consortium.

M. Carmen, P. Felt, R. Haertel, D. Lonsdale, P. McClanahan, O. Merkling, E. Ringger, and K. Seppi. 2010. *Tag dictionaries accelerate manual annotation*. In Proceedings of the Seventh International Conference on Language Resources and Evaluation. European Language Resources Association, Valletta, Malta.

T. Caselli, I. Chiari, A. Gangemi, E. Jezek, A. Oltramari, G. Vetere, L. Vieu, and F. M. Zanzotto. 2014. *Senso Comune as a knowledge base of Italian language: the resource and its development*. In Italian Conference on Computational Linguistics 2014, volume 1, pp. 93–97. Pisa University Press.

C. Chelba and A. Acero. 2004. *Adaptation of maximum entropy capitalizer: Little data can help a lot*. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pp. 285–292. Association for Computational Linguistics, Barcelona, Spain.

G. Chrupała. 2006. *Simple data-driven context-sensitive lemmatization*. Procesamiento del Lenguaje Natural, volume 37.

G. Chrupała. 2008. *Towards a machine-learning achitecture for lexical functional grammar parsing*. Ph.D. thesis.

G. Chrupała, G. Dinu, and J. van Genabith. 2008. *Learning morphology with Morfette*. In Proceedings of the Sixth International Conference on Language Resources and Evaluation. European Language Resources Association, Marrakech, Morocco.

R. Clawson and W. Barrett. 2015. *Intelligent indexing: A semi-automated, trainable system for field labeling*. In Document Recognition and Retrieval XXII, Proceedings of SPIE-IS&T Electronic Imaging, volume 9402. SPIE-IS&T.

M. Collins. 2002. *Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms*. In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, pp. 1–8. Association for Computational Linguistics, Stroudsburg, PA, USA.

R. Cotterell, T. Müller, A. Fraser, and H. Schütze. 2015. *Labeled morphological segmentation with semi-Markov models*. In Proceedings of the Nineteenth Conference on Computational Natural Language Learning, pp. 164–174. Association for Computational Linguistics, Beijing, China.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. *Online passive-aggressive algorithms*. Journal of Machine Learning Research, volume 7:pp. 551–585.

K. Crammer and Y. Singer. 2003. *Ultraconservative online algorithms for multiclass problems*. Journal of Machine Learning Research, volume 3:pp. 951–991.

A. Culotta, T. Kristjansson, A. McCallum, and P. Viola. 2006. *Corrective feedback and persistent learning for information extraction*. Artificial Intelligence, volume 170(14–15):pp. 1101 – 1122.

A. Culotta and A. McCallum. 2005. *Reducing labeling effort for structured prediction tasks*. In Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2, pp. 746–751. AAAI Press.

H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters. 2011. *Text Processing with GATE*. Gateway Press CA.

K. Darwish, A. Abdelali, and H. Mubarak. 2014. *Using stem-templates to improve Arabic POS and gender/number tagging*. In Proceedings of the Ninth International Conference on Language Resources and Evaluation. European Language Resources Association, Reykjavik, Iceland.

G. Druck and A. McCallum. 2011. *Toward interactive training and evaluation*. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management. Association for Computing Machinery, New York, NY, USA.

K. Dukes. 2013. *Statistical parsing by machine learning from a classical Arabic treebank*. Ph.D. thesis, PhD Thesis, University of Leeds.

K. Dukes and N. Habash. 2010. *Morphological annotation of Quranic Arabic*. In Proceedings of the Seventh International Conference on Language Resources and Evaluation. European Language Resources Association, Valletta, Malta.

J. A. Fails and D. R. Olsen, Jr. 2003. *Interactive machine learning*. In Proceedings of the 8th International Conference on Intelligent User Interfaces, pp. 39–45. Assoication for Computing Machinery, New York, NY, USA.

P. Felt, O. Merkling, M. Carmen, E. Ringger, W. Lemmon, K. Seppi, and R. Haertel. 2010. *CCASH: A web application framework for efficient, distributed language resource development*. In Proceedings of the Seventh International Conference on Language Resources and Evaluation.

P. Felt, E. K. Ringger, K. Seppi, K. S. Heal, R. A. Haertel, and D. Lonsdale. 2013. *Evaluating machine-assisted annotation in under-resourced settings*. Language Resources and Evaluation.

N. Habash. 2007. *Arabic morphological representations for machine translation*. In Arabic Computational Morphology, pp. 263–285. Springer Netherlands.

N. Habash and O. Rambow. 2005. *Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop*. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, pp. 573–580. Association for Computation Linguistics, Stroudsburg, PA.

R. Haertel, P. Felt, E. K. Ringger, and K. Seppi. 2010a. *Parallel active learning: Eliminating wait time with minimal staleness*. In Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing, pp. 33–41. Association for Computational Linguistics, Los Angeles, CA.

R. Haertel, P. McClanahan, and E. K. Ringger. 2010b. *Automatic diacritization for low-resource languages using a hybrid word and consonant CMM*. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 519–527. Association for Computational Linguistics, Los Angeles, CA.

R. A. Haertel. 2013. *Practical cost-conscious active learning for data annotation in annotator-initiated environments*. Ph.D. thesis, Brigham Young University.

L. Huang, S. Fayong, and Y. Guo. 2012. *Structured perceptron with inexact search*. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 142–151. Association for Computational Linguistics, Stroudsburg, PA, USA.

S. Jiampojamarn, A. Bhargava, Q. Dou, K. Dwyer, and G. Kondrak. 2009. *DirecTL: a language independent approach to transliteration*. In Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration, pp. 28–31. Association for Computational Linguistics, Suntec, Singapore.

S. Jiampojamarn, C. Cherry, and G. Kondrak. 2008. *Joint processing and discriminative training for letter-to-phoneme conversion*. In Proceedings of ACL-08: HLT, pp. 905–913. Association for Computational Linguistics, Columbus, Ohio.

S. Jiampojamarn, C. Cherry, and G. Kondrak. 2010. *Integrating joint n-gram features into a discriminative training framework*. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 697–700. Association for Computational Linguistics, Los Angeles, California.

S. Jiampojamarn, G. Kondrak, and T. Sherif. 2007. *Applying many-to-many alignments and hidden Markov models to letter-to-phoneme conversion*. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference, pp. 372–379. Association for Computational Linguistics, Rochester, New York.

N. Kaji, Y. Fujiwara, N. Yoshinaga, and M. Kitsuregawa. 2010. *Efficient staggered decoding for sequence labeling*. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 485–494. Association for Computational Linguistics, Stroudsburg, PA, USA.

A. Kilgarriff. 2005. *Linking dictionary and corpus*. In Proceedings of ASIALEX 2005. Singapore.

A. Kilgarriff and M. Rundell. 2002. *Lexical profiling software and its lexicographic applications - a case study*. In Proceedings of the 10th EURALEX International Congress, pp. 807–818. Center for Sprogteknologi, København, Denmark.

G. A. Kiraz. 1994. *Automatic concordance generation of Syriac texts*. Orientalia Christiana Analecta, volume 247:pp. 461–475.

G. A. Kiraz. 2000. *Multitiered nonlinear morphology using multitape finite automata: A case study on Syriac and Arabic*. Computational Linguistics, volume 26(1):pp. 77–105.

T. Kristjansson, A. Culotta, P. Viola, and A. McCallum. 2004. *Interactive information extraction with constrained conditional random fields*. In Proceedings of the 19th National Conference on Artificial Intelligence. AAAI Press.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. 2001. *Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data*. In Proceedings of the Eighteenth International Conference on Machine Learning, pp. 282–289. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

E. W. Lane. 1863. *Arabic-English Lexicon*. Williams & Norgate, London.

V. I. Levenshtein. 1966. *Binary codes capable of correcting deletions, insertions, and reversals*. In Soviet Physics—Doklady, volume 10, pp. 707–710.

L. Lindgren. 2011. *dkrMorph: A Syriac morphological analyzer*. Bachelor's thesis, Uppsala University, Department of Informatics and Media.

A. McCallum, D. Freitag, and F. C. N. Pereira. 2000. *Maximum entropy Markov models for information extraction and segmentation*. In Proceedings of the Seventeenth International Conference on Machine Learning, pp. 591–598. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

P. McClanahan. 2010. *A probabilistic morphological analyzer for Syriac*. Masters thesis, Brigham Young University, Department of Computer Science.

P. McClanahan, G. Busby, R. Haertel, K. Heal, D. Lonsdale, K. Seppi, and E. Ringger. 2010. *A probabilistic morphological analyzer for Syriac*. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pp. 810–820. Association for Computational Linguistics, Cambridge, MA.

R. Mihalcea and A. Csomai. 2007. *Wikify!: Linking documents to encyclopedic knowledge*. In Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, pp. 233–242. Association for Computing Machinery, New York, NY, USA.

R. Navigli. 2009. *Word sense disambiguation: A survey*. ACM Computing Surveys, volume 41(2):pp. 10:1–10:69.

S. B. Needleman and C. D. Wunsch. 1970. *A general method applicable to the search for similarities in the amino acid sequence of two proteins*. Journal of Molecular Biology, volume 48(3):pp. 443–453.

J. Payne Smith. 1903. *A compendious Syriac dictionary founded upon the thesaurus Syriacus of R. Payne Smith*.

A. J. Quinn and B. B. Bederson. 2011. *Human computation: A survey and taxonomy of a growing field*. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1403–1412. Association for Computing Machinery, New York, NY, USA.

E. Ringger, M. Carmen, R. Haertel, K. Seppi, D. Lonsdale, P. McClanahan, J. Carroll, and N. Ellison. 2008. *Assessing the costs of machine-assisted corpus annotation through a user study*. In Proceedings of the Sixth International Conference on Language Resources and Evaluation, pp. 3318–3324. European Language Resources Association, Marrakech, Morocco.

E. S. Ristad and P. N. Yianilos. 1998. *Learning string-edit distance*. IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 20(5):pp. 522–532.

D. Seddah, G. Chrupała, O. Çetinoğlu, J. van Genabith, and M. Candito. 2010. *Lemmatization and lexicalized statistical parsing of morphologically rich languages: The case of french*. In Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages, pp. 85–93. Association for Computational Linguistics, Stroudsburg, PA, USA.

B. Settles. 2011. *Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances*. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Stroudsburg, PA, USA.

B. Settles. 2012. *Active Learning: Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers.

B. Settles and X. Zhu. 2012. *Behavioral factors in interactive training of text classifiers*. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 563–567. Association for Computational Linguistics, Montréal, Canada.

P. Shivaswamy and T. Joachims. 2012. *Online structured prediction via coactive learning*. In Proceedings of the 29th International Conference on Machine Learning, pp. 1431–1438. Omnipress, New York, NY, USA.

A. Sokolov, S. Riezler, and S. B. Cohen. 2015. *A coactive learning view of online structured prediction in statistical machine translation*. In Proceedings of the Nineteenth Conference on Computational Natural Language Learning, pp. 1–11. Association for Computational Linguistics, Beijing, China.

P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii. 2012. *brat: a web-based tool for NLP-assisted text annotation*. In Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, Avignon, France.

M. Stonebraker, U. Çetintemel, and S. Zdonik. 2005. *The 8 requirements of real-time stream processing*. SIGMOD Record, volume 34(4):pp. 42–47.

C. Sutton and A. McCallum. 2006. *An Introduction to Conditional Random Fields for Relational Learning*, chapter in Introduction to Statistical Relational Learning. MIT Press.

C. Sutton and A. McCallum. 2012. *An introduction to conditional random fields*. Foundations and Trends in Machine Learning, volume 4(4):p. 267–373.

J. J. Thomas and K. A. Cook. 2005. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society Press.

K. Tomanek and U. Hahn. 2009. *Semi-supervised active learning for sequence labeling*. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pp. 1039–1047. Association for Computational Linguistics, Suntec, Singapore.

K. Toutanova and C. Cherry. 2009. *A global model for joint lemmatization and part-of-speech prediction*. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pp. 486–494. Association for Computational Linguistics, Suntec, Singapore.

T. T. Truyen, H. H. Bui, D. Q. Phung, and S. Venkatesh. 2008. *Learning discriminative sequence models from partially labelled data for activity recognition*. In Tu-Bao Ho and Zhi-Hua Zhou (editors), PRICAI

2008: Trends in Artificial Intelligence, volume 5351 of *Lecture Notes in Computer Science*, pp. 903–912. Springer Berlin Heidelberg.

Y. Tsuboi, H. Kashima, S. Mori, H. Oda, and Y. Matsumoto. 2008. *Training conditional random fields using incomplete annotations*. In Proceedings of the 22nd International Conference on Computational Linguistics, pp. 897–904. Coling 2008 Organizing Committee, Manchester, UK.

O. Täckström, D. Das, S. Petrov, R. McDonald, and J. Nivre. 2013. *Token and type constraints for cross-lingual part-of-speech tagging*. Transactions of the Association for Computational Linguistics, volume 1:pp. 1–12.

N. J. Venhuizen, V. Basile, K. Evang, and J. Bos. 2013. *Gamification for word sense labeling*. In Proceedings of the 10th International Conference on Computational Semantics – Short Papers, pp. 397–403. Association for Computational Linguistics, Potsdam, Germany.

A. J. Viterbi. 1967. *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm*. IEEE Transactions on Information Theory, volume 13(2):pp. 260–269.

L. von Ahn. 2006. *Games with a purpose*. Computer, volume 39(6):pp. 92–94.

L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. 2008. *reCAPTCHA: Human-based character recognition via web security measures*. Science, volume 321(5895):pp. 1465–1468.

S. M. Yimam, R. Eckart de Castilho, I. Gurevych, and C. Biemann. 2014. *Automatic annotation suggestions and custom annotation layers in WebAnno*. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. System Demonstrations, pp. 91–96. Association for Computational Linguistics, Stroudsburg, PA 18360, USA.

M.-C. Yuen, I. King, and K.-S. Leung. 2011. *A survey of crowdsourcing systems*. In Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom), pp. 766–773.