Parsing Early Modern English for Linguistic Search

Seth Kulick and **Neville Ryant**
Linguistic Data Consortium
University of Pennsylvania
{skulick,nryant}@ldc.upenn.edu

Beatrice Santorini
Linguistics Dept.
University of Pennsylvania
beatrice@sas.upenn.edu

Abstract

This work addresses the question of whether the output of a state-of-the-art parser is accurate enough to support research in theoretical linguistics. In order to build reliable models of syntactic change, we aim to eventually parse the 1.5-billion-word Early English Books Online (EEBO) corpus. But since EEBO is not yet parsed, we begin by constructing and testing a parser on the 1.7-million-word Penn-Helsinki Parsed Corpus of Early Modern English (PPCEME). In order to obtain robust results, we define an 8-fold split on PPCEME. We then evaluate the parser with evalb and, more relevantly for us, with a task-specific metric - namely, its accuracy in parsing 6 sentence types necessary to track the rise of auxiliary *do* (as in *They did not come* vs. its historical precursor *They came not*). Retrieving the relevant sentences from the gold and test versions with CorpusSearch queries (Randall, 2010), we find that the parser's accuracy promises to be sufficient for our purposes. A remaining concern is the variability of the output, which we plan to address with three pieces of future work sketched in the conclusion.

1 Introduction

The Penn-Helsinki Parsed Corpus of Early Modern English (PPCEME) (Kroch et al., 2004) consists of over 1.7 million words of text from 1500 to 1712, manually annotated for phrase structure. It belongs to a family of treebanks of historical English (Taylor et al., 2003; Kroch et al., 2000b; Taylor et al., 2006; Kroch et al., 2016) and other languages (Wallenberg et al., 2011; Galves et al., 2017; Martineau et al., 2021; Kroch and Santorini, 2021) with a shared annotation philosophy and similar guidelines across languages, which form the basis for reproducible studies of syntactic change (Kroch et al., 2000a; Ecay, 2015; Wallenberg, 2016; Galves, 2020; Wallenberg et al., 2021).

While all of these corpora are relatively large for manually annotated corpora, there are important limits on their usefulness - notably, the fact that even relatively common phenomena still occur too rarely to support reliable statistical models of how they change over time. We therefore wish to parse and search the much larger corpora that are becoming publicly available. For instance, with its 1.5 billion words of text from 1475 to 1700, the Early English Books Online (EEBO) corpus (Text Creation Partnership, 2019) dwarfs PPCEME. However, its potential as a resource for linguistic research remains unrealized because it is not linguistically annotated and its size renders manual annotation infeasible. Our eventual goal is therefore to parse EEBO automatically.

This paper reports on a first step in that direction - namely, building a parser whose accuracy we can evaluate on the gold standard provided by PPCEME. For our purposes, the standard evaluation metric, evalb (Sekine and Collins, 2008), is not specific enough. Evaluation measures based on joint effects of parser output with other factors are also inappropriate, since retrieving the sentence types of interest to us is a direct function of the parse, without any intervening processing. It is clear that the most useful evaluation metric for our purposes involves scoring the retrieval of the diagnostic sentence types. Here, we report on negative declarative sentences, on negative imperatives, and on direct questions, each in two variants. The first variants are the ones that were dominant in 1500 (*They drank not the ale, Drink not the ale, Drank they the ale?*), and the second ones are their modern counterparts, which had become dominant by 1700 (*They did not drink the ale, Do not drink the ale, Did they drink the ale?*). We choose these sentence types because we hope that large datasets like EEBO will eventually allow us to decide between different conceptual models of the change - specifically, competition (Kroch, 1989; Zimmermann, 2017) versus drift (Karjus, 2020).

The remainder of the paper is structured as follows. Section 2 discusses some features of PPCEME's source material and annotation that present challenges for state-of-the-art parsers, especially as compared to more widely used treebanks such as the Penn Treebank (PTB) (Marcus et al., 1993). Sections 3 and 4 describe our cross-validation split of PPCEME for evaluating the parser and our use of EEBO to create contextualized word embeddings for the parser. Section 5 presents the parser model, along with results based on evalb, which we include for general comparability beyond our task-specific evaluation metric. Section 6 illustrates the diagnostic sentence types and the queries that we use for retrieving them, which are formulated in the CorpusSearch query language (Randall, 2010). Section 7 presents the results from the task-specific evaluation, and Section 8 summarizes with an eye towards future work.

2 PPCEME Issues

PPCEME differs from PTB in several important ways, making it an excellent test case for domain adaptation of modern parsing technology. However, there has been relatively little work in the NLP community using PPCEME and its sister corpora, the Penn Parsed Corpus of Middle English, 2nd edition (PPCME2) and the Penn Parsed Corpus of Modern British English (PPCMBE).[1] Kulick et al. (2014) describe parsing PPCMBE, while Moon and Baldridge (2007) and Yang and Eisenstein (2016) focus on POS-tagging (the former on PPCME2, and the latter on PPCEME and PPCMBE).

In addition to the nonstandard orthography and the different and variable syntax of the source material, PPCEME is annotated according to guidelines arising in part from its purpose for linguistic research that require explicit consideration.

2.1 PPCEME Part-of-Speech Tags

2.1.1 Complex Tags

Although we generally attempt to avoid modifying the existing annotation, PPCEME's very large set of POS tags (N = 353) requires trimming to a computationally more tractable size.

Of the 353 tags just mentioned, 213 are complex tags intended to facilitate tracking changes in orthographic conventions over time - for instance, the development of `(ADJ gentle) (NS men)` to `(ADJ+NS gentlemen)`. Since these changes are irrelevant for present purposes, we prune such tags in accordance with the Righthand Head Rule, yielding `(NS gentlemen)`.[2] Certain rare cases, such as `(WPRO+ADV+ADV whatsoever)` or `(Q+BEP+PRO albeit)`, are exceptions to the Righthand Head Rule. In such cases, the best simple tag is sometimes the leftmost tag and sometimes another tag entirely (`(WPRO whatsoever)`, `(P albeit)`). We simply ignore this complication on the grounds that these cases are a small subset of the complex tags, which themselves are used for only about 1% of the words in the corpus. After pruning and some other minor changes discussed in Appendix A, 85 POS tags remain.

2.1.2 Distinctions among Verb Classes

PTB makes no distinction between main verbs and the auxiliary verbs *be*, *do* and *have*, but this distinction is vital for us, since it is exactly the syntax of main (but not auxiliary) verbs that changes over the course of Early Modern English. In fact, even among the verbs with auxiliary uses, we need to distinguish *do* from the other auxiliaries in order to track the rise of auxiliary *do*. For this reason, we do not follow Yang and Eisenstein (2016) in mapping the PPCEME tags for verbs to the smaller set used in PTB.

2.2 PPCEME Phrase Structure

2.2.1 Function Tags

In phrase-structure treebanks, function tags can be appended to syntactic category labels in order to provide information about a constituent's grammatical or semantic role. The PTB uses 20 function tags in this way, while exploiting structural differences to distinguish other constituent roles. By contrast, PPCEME relies on function tags uniformly, largely because it has neither base NPs or VPs. As a result, PPCEME's set of function tags is larger than PTB's. Omitting a few rare types, we use 31 in the work reported below.[3] The following tree illustrates PPCEME's use of function tags to encode central grammatical roles. The subject and indirect object are sisters, but distinguished by the

[1] These two corpora and PPCEME are collected in Kroch (2020).

[2] Yang and Eisenstein (2016) simplify the complex tags for the same reason as we do, but keep the leftmost tag, which for English is incorrect in the general case.

[3] See Appendix B for the details, along with some information on function tag frequency.

function tags SBJ and OB2, respectively. MAT and SUB on the two IPs identify the higher one as a matrix clause and the lower one as a subordinate clause. Finally, THT indicates that the CP is a *that* complement clause (rather than, say, a relative or adverbial clause).

```
(IP-MAT (CONJ and)
        (NP-SBJ (D the) (N schereffe))
        (VBD shewed)
        (NP-OB2 (PRO$ my) (N servant))
        (CP-THT (C that)
                (IP-SUB ...)))
```

There has been some work on recovering function tags in PTB (Blaheta and Charniak, 2000; Blaheta, 2003; Gabbard et al., 2006; Merlo and Musillo, 2005), but overall they have received only limited attention. We are not aware of any work to recover the function tags in the historical corpora. Given the centrality of certain function tags (notably, SBJ) for retrieving the sentence types of interest to us, we are constrained to include them in the parsing model.

2.2.2 Empty Categories

PPCEME indicates discontinuous dependencies by means of empty categories that are coindexed with a displaced constituent. Following common NLP practice, we remove both the empty categories and the co-indexing from the parser training material, and thus from the parser output. This simplifies the parsing model, and for present purposes, the absence of empty categories is irrelevant. However, if we wish to include linguistic queries in future work that make reference to empty categories, as is necessary in the general case, the parsing model will need to be augmented appropriately.

3 Cross-validation Splits

Parsing work relies on train/dev/test splits of the source material used for training and evaluation. Recently, concerns have been raised over the validity of inferences drawn from static train/dev/test splits; for instance, see Gorman and Bedrick (2019), who evaluate the consistency of rankings of POS taggers across 20 random splits of the WSJ section of PTB. For us, this issue is particularly pressing because PPCEME contains relatively few individual source texts, thus increasing the chance that a single particularly difficult or non-representative source text will greatly skew performance on the dev/test partitions. Even more seriously, certain constructions might be completely absent from one particular split. This is of particular concern to us because direct questions, though common in ordinary conversation, are rare or completely absent in many written genres. We return to this point in Section 7.2.2.

We therefore define an 8-fold cross-validation split, with each component split roughly matching the 90%-5%-5% distribution in the standard single PTB split. Within each partition (train, dev, test) of a split, we attempted to equally represent (in terms of equal word counts) each of PPCEME's three time periods, as indicated by "e1", "e2", and "e3" in the filenames. Given our eventual goal of parsing all of EEBO, which encompasses all of these time periods, this step is necessary in order to adequately predict performance on that corpus.[4] Finally, in cases where PPCEME distributes a single source text over several annotated files, we were careful to assign all such files to the same partition. As PPCEME contains 448 annotated files, but only 232 distinct source texts, this greatly constrained how we could define the partitions. Nevertheless, we succeeded in including 209 (90%) of the 232 source texts in either a dev or test partition of one of the 8 splits. For more details on the split definitions, see Appendix C.

4 ELMo Embeddings Trained on EEBO

In recent years, contextualized word embeddings such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) have driven significant improvements on downstream NLP tasks, including POS tagging and parsing. Due to the significant overhead involved in training these representations, researchers often use pretrained models distributed by large companies, sometimes fine-tuned to the domain of interest. Although this often produces perfectly satisfactory results, in cases of significant mismatch between a test domain and standard training domains - usually sources such as text scraped from Wikipedia, BooksCorpus (Zhu et al., 2015), and news text from Common Crawl (Nagel, 2016) - pretraining on the novel domain yields significant improvements (Lee et al., 2019; Beltagy et al., 2019; Jin et al., 2019).

Because of the orthographic and syntactic differences between Early Modern English and contemporary English mentioned in Section 2, our current

[4] By contrast, Yang and Eisenstein (2016), split PPCEME into thirds by time period (rather than across time periods) for the different purpose of studying domain adaptation.

work involves exactly such a mismatch, and so we pretrained ELMo embeddings on EEBO.[5]

We used the same model configuration as Peters et al. (2018) for 11 epochs[6] using all of EEBO. We then integrated the resulting embeddings, which have 1,024 dimensions, into the parser model, as discussed in Section 5. Here, we describe some main aspects of creating the embeddings, which we will make public. See Appendix D for further details.

4.1 Text Extraction, Normalization, and Tokenization

EEBO's XML files contain a great deal of metadata and markup in addition to the text. For each file, we extracted the core source information (title, author, date) and kept the text within `<P>` tags, which gives at least a rough sense of the document divisions. Following Ecay (2015, pp. 105-6), we excluded some metadata and other material embedded in the text. We also adopted his handling of `GAP` tags for OCR errors, which consists of mapping these tags to word-internal bullet characters - e.g., `Eccl•siasticall`.

After normalizing the extracted text with Unicode NFC form in order to eliminate spurious surface differences between tokens, we tokenized the EEBO text in accordance with PPCEME's tokenization guidelines as best we could:

1. Possessive morphemes are not separated from their host (e.g., `Queen's`) (unlike in PTB).

2. Punctuation is separated except in the case of abbreviations (e.g., `Mr.`), token-internal hyphens (e.g., `Fitz-Morris`), or certain special cases (e.g., `&c`).

3. Roman numerals can include leading, internal, or trailing periods (e.g., `.xiiii.C.`).

PPCEME tokenization is straightforward in principle, but the non-standardized nature of the historical material raises various difficulties. For instance, it is easy to tell that the elided article `th'` should be split off (e.g., `th'exchaung` is tokenized as `th' exchaung`). But when the apostrophe is missing, the status of `th` is unclear (e.g., `thafternoone` is tokenized as `th afternoone`, but `thynkyth` remains a single token). Another example of pervasive ambiguity is `its` and `it's`; in PPCEME, these forms were tokenized manually as one token or two, depending on whether the spelling represents the possessive form of the pronoun *it* or the contracted form of *it is*. Since EEBO's size rules out manual processing, we resolved such ambiguities by defaulting to the more common case. In the above examples, this resulted in splitting the variants with apostrophes and not splitting the ones without.[7]

5 Model and Evaluation

5.1 Parser Architecture

We use the parser model of Kitaev et al. (2019), which represents a constituency tree T as a set of labeled spans (i, j, l), where i and j are a span's beginning and ending positions and l is its label. Each tree is assigned a score $s(T)$, which is decomposed as a sum of per-span scores:

$$s(T) = \sum_{(i,j,l)\in T} s(i, j, l) \qquad (1)$$

The per-span scores $s(i, j, l)$ themselves are assigned using a neural network that takes a sequence of per-word embeddings as input, processes these embeddings using a transformer-based encoder (Vaswani et al., 2017), and produces a span score from an MLP classifier (Stern et al., 2017). The highest-scoring valid tree is then found using a variant of the CKY algorithm. POS tags are recovered using a separate classifier operating on top of the encoder output, which is jointly optimized with the span classifier. For more details, see Kitaev and Klein (2018). As already mentioned in Section 4, we use ELMo embeddings pre-trained on EEBO.

Our implementation is based on version 0.2.0 of the Berkeley Neural Parser[8] modified to accept ELMo.[9] We train each of the 8 models (one for each cross-validation split) for 50 epochs, using the evalb score on the dev section as our criterion for

[5] Space constraints prevent us from presenting full details here, but we find that using ELMo embeddings trained on EEBO improves evalb scores by about 2 points over the standard ELMo embeddings trained on modern English and still by about 0.5 points over BERT embeddings trained on modern English. At present, we lack the computational resources for the obvious next step of pretraining BERT embeddings on EEBO, but we are pursuing access to them.

[6] This corresponds to 2 weeks of training using 4 GTX 1080 GPUs.

[7] Future work could consider a joint tokenization-POS-tagging model.

[8] https://github.com/nikitakit/self-attentive-parser

[9] These modifications and other relevant software are available at https://github.com/skulick/emeparse.

	Parser	Part-of-Speech
dev	90.89 (1.8)	98.14 (0.7)
test	90.53 (0.7)	98.30 (0.4)

Table 1: Cross-validation Parser and Part-of-Speech Results. Each result is the mean for the relevant partition (dev or test) over the 8 splits, with the standard deviation in parentheses.

saving as the best model. For more details regarding training and hyperparameters, see Appendix E.

5.2 Function Tags

Following the approach of Gabbard et al. (2006) to function tag recovery, we do not delete function tags in preprocessing, and so nonterminals like NP-SBJ are treated as atomic units. Since the decision whether to delete is part of the preprocessing, this approach does not require modification to the parser.

5.3 Evalb Results

Table 1 gives our parsing and part-of-speech results by the standard NLP measures, combined over the 8 cross-validation splits, as scored by evalb (matching brackets for the parsing score and POS accuracy for the tagging score).[10]

The evalb parsing score falls within the general range of parsing scores for PTB, though a few points lower. As Kulick et al. (2014) point out, all of the English historical corpora lack certain brackets present in PTB (base NPs and VPs) that are relatively "easy to get", and this tends to adversely affect their parsing scores. Specifically, Kulick et al. (2014) find the f1 score for PPCMBE to be lower than for PTB by about 2 points, and we would expect that effect to carry over to PPCEME.[11]

6 Diagnostic Sentence Types and Query-based Retrieval

Having obtained a rough idea of the parser's performance from the evalb scores, we now turn to the question of greater interest to us - the evaluation of the parser in task-specific terms. Recall that we wish to identify certain sentence types that allow us to track the rise of auxiliary *do* over the course of Early Modern English. For expository reasons, we present these sentence types in reverse chronological order.[12]

6.1 Sentence Types with Auxiliary *Do*

Modern English is unusual in requiring the auxiliary verb *do* in certain sentence types, notably in negative declarative sentences, in negative imperatives, and in all direct questions (whether positive or negative).

`Do-not-decl.` In negative declarative sentences, the main verb appears in uninflected form. Such sentences also contain auxiliary *do* in either the present or past tense, and the negative marker *not* appears between the auxiliary and the main verb.

```
(IP-SUB (NP-SBJ (PRO they))
        (DOP do)
        (NEG not)
        (NP-MSR (Q much))
        (VB minde)
        (NP-OB1 (PRO them))
```

As the above example shows, the IP in this sentence type (and also its historical counterpart) can be either an independent matrix (MAT) clause or, as here, a subordinate (SUB) clause.

`Do-not-imp.` Negative imperatives are analogous, except for the IMP function tag on IP, and the imperative POS tag (DOI) on the auxiliary.

```
(IP-IMP (PP (P For)
            (NP (NPR$ God's)
                (N sake)))
       (DOI do)
       (NEG not)
       (VB overlay)
       (NP-OB1 (PRO me))
       (PP (P with)
           (NP (ADJ superfluous)
               (N Matter)))
    (. .))
```

`Do-sbj.` Finally, in direct questions, auxiliary *do* precedes the subject instead of following it, as in declaratives. This inversion occurs in both positive and negative questions, and so retrieving this sentence type relies crucially on the parser correctly identifying the subject via the SBJ function tag. In the following example, note that the annotation guidelines for PPCEME require direct questions to

[10] Evalb removes tokens (punctuation) from consideration based on their POS tags, and since our model predicts POS tags, this can result in inconsistent sentence lengths for the gold and parsed trees if there are POS tag errors, resulting in "Error" sentences in the evalb output. We therefore use the modified evalb supplied with the Berkeley parser, due to Seddah et al. (2014), which does not delete any words, so that any POS tag differences have no effect on sentence length.

[11] For some discussion of function tag accuracy from an NLP perspective, see Appendix F.

[12] We are concerned only with sentences without modal verbs (*can*, *will*, etc.), aspectual auxiliaries *have* and *be*, or main verb *be*; sentences containing these elements were not affected by the change.

be annotated as CP-QUE-MAT immediately dominating IP-SUB. In this context, the IP-SUB is understood as part of the direct question rather than an ordinary subordinate clause.

```
(CP-QUE-MAT (WADVP (WADV How))
    (IP-SUB (DOP do's)
            (NP-SBJ (D this) (N Sute))
            (VB fit)
            (NP-OB1 (PRO me)))
    (NP-VOC (NPR Dauy))
    (. ?))
```

6.2 Sentence Types Without Auxiliary *Do*

We now illustrate the historical precursors of the modern sentence types just discussed. In all 3 old forms, it is the main verb (rather than auxiliary *do*) that appears in a past or present tense form, and it occupies the same position as auxiliary *do*. Thus, we have negative declarative sentences (`verb-decl-not`) like:

```
(IP-SUB (NP-SBJ (PRO I))
        (VBD sent)
        (NEG not)
        (PP (P to)
             (NP (PRO you))))
```

negative imperatives (`verb-not-imp`) like:

```
(IP-IMP (VBI let)
        (NEG not)
        (IP-INF (NP-SBJ (D that))
                (VB hurt)
                (NP-OB1 (PRO me)))
    (. .))
```

and questions (`verb-sbj`) like:

```
(CP-QUE-MAT
    (WADVP (WADV When))
    (IP-SUB (VBP comes)
            (NP-SBJ (PRO$ your)
                    (N Taylor))
            (ADVP-DIR (ADV hither)))
    (. ?))
```

6.3 Sample CorpusSearch Query

In order to retrieve the 6 diagnostic sentence types, we formulate queries in CorpusSearch (Randall, 2010), a query language for querying, editing, and coding tree structures. Each query is a sequence of boolean conditions on the parser output. For instance, the following query retrieves direct questions with auxiliary *do* (`do-sbj`).

```
    (CP-QUE-MAT* iDoms IP-SUB*)
AND (IP-SUB* iDoms DOD|DOP)
AND (IP-SUB* iDoms NP-SBJ*)
AND (IP-SUB* iDoms DO|VB)
AND (DOD|DOP precedes NP-SBJ*)
AND (NP-SBJ* precedes DO|VB)
```

The asterisks on the labels allow the query to match tokens with further trailing function tags (say, -SPE to indicate direct speech or -RSP for resumptive subjects). In concluding this section, we draw the reader's attention to the fact that our queries are all formulated assuming that the parser has constructed the relevant clause boundaries correctly. In Section 7.2.1, we discuss an attempt to improve parser performance by allowing structures without IP-SUB or with a recursive IP-SUB to count as matches.

7 Query-Based Results and Analysis

7.1 Results

We evaluated the parser in task-specific terms as follows. For each split, we (1) trained the parser on that split's training section and (2) parsed the split's dev and test sections. We then (3) ran 6 CorpusSearch queries, one for each of the diagnostic sentence types just presented, over the parsed sections. Cases where the queries retrieved "hits" in both the gold and the parsed tree were matches. Hits in the gold, but not the parsed tree, were classified as misses. The converse case of hits in the parsed tree, but not in the gold, were false alarms. From the results for these categories, we calculated the recall, precision, and f-measure for each split.

We calculated the mean and standard deviation of the recall, precision and f-measure over each split's dev section and over each split's test section. These results are shown in Table 2, with the associated standard deviations in parentheses. For each query, we also include the number (#) of hits in the gold version of the trees. These results are analogous to the cross-validation results using evalb in Table 1, but with the individual cross-validated query-based scores instead of the evalb metric.

7.2 Analysis

The overall f1 scores based on the queries are for the most part high enough for the overall project to remain promising. We neither expect complete parser accuracy, nor do we require it, since we can include an estimated error rate in any statistical models that we build.

However, the results exhibit a degree of variability that calls for investigation. The standard deviations are all higher than for the evalb results in Table 1, even for negative declarative sentences (the best case). This follows from the relative sparseness of the diagnostic structures in the corpus, as

	dev				test			
query	#	recall	prec	f1	#	recall	prec	f1
Negative declarative sentences								
do-not-decl	338	94.97 (3.7)	98.92 (1.7)	96.86 (1.9)	405	93.39 (4.3)	98.40 (2.3)	95.74 (1.9)
verb-not-decl	717	93.79 (4.9)	93.71 (3.3)	93.72 (3.8)	653	92.94 (4.0)	93.42 (3.4)	93.10 (2.5)
Negative imperative sentences								
do-not-imp	41	72.37 (45.3)	71.72 (44.8)	71.83 (44.7)	23	77.71 (34.1)	87.50 (35.4)	81.83 (34.0)
verb-not-imp	120	86.03 (10.4)	91.91 (7.4)	88.22 (4.2)	142	75.61 (20.0)	92.10 (6.8)	82.24 (14.9)
Questions								
do-sbj	564	89.29 (6.3)	98.32 (2.3)	93.47 (3.8)	329	84.48 (16.5)	93.75 (17.7)	86.57 (13.0)
verb-sbj	387	81.01 (13.2)	95.39 (3.8)	87.10 (8.1)	190	69.68 (19.7)	87.29 (10.9)	75.67 (14.4)
Augmented questions								
do-sbj+	564	92.23 (5.8)	98.36 (2.3)	95.10 (3.4)	329	85.92 (16.2)	93.75 (17.7)	87.44 (13.3)
verb-sbj+	387	84.16 (11.9)	94.00 (5.1)	88.35 (7.0)	190	74.39 (19.6)	83.10 (11.2)	76.26 (13.0)

Table 2: Query-based Results for the Dev and Test Sections. The first 6 sentence types are illustrated in Section 6. Augmented questions are discussed in Section 7.2.1.

compared to the much higher number of brackets evaluated by evalb. We turn now to two dimensions of this variability.

7.2.1 Recall vs. Precision and Parser Errors

In general, the recall results are lower than the precision results across all sentence types. By examining recall errors in the dev section, we have identified two of the more frequent error types.[13]

The first is an unfortunate tendency for the parser to produce nonsensical structures rather than to build parenthetical clauses. For example, for this gold question:

```
(CP-QUE-MAT
   (IP-SUB (IP-MAT-PRN (NP-SBJ (PRO I))
                       (VBP pray)
                       (NP-OB2 (PRO you)))
           (VBP speketh)
           (NP-SBJ (PRO he))
           (PP (P vnto)
               (NP (PRO vs)))))
```

the parser generates a flat structure with two subjects and two finite verbs, which is neither reasonable nor found in the training data (nor, for that matter, in the entire corpus).[14]

```
(CP-QUE-MAT
    (IP-SUB (NP-SBJ (PRO I))
            (VBP pray)
            (NP-OB2 (PRO you))
            (VBP speketh)
            (NP-SBJ (PRO he))
            (PP (P vnto)
                (NP (PRO vs)))))
```

A second problem that became apparent in connection with questions, both with and without auxiliary *do*, is that the parser sometimes violates the PPCEME's annotation guidelines by either omitting IP-SUB under CP-QUE-MAT or adding a redundant one. For example, instead of the gold

```
(CP-QUE-MAT (WNP (WPRO What)
            (IP-SUB (ADVP (ADV then))
                    (VBP think)
                    (NP-SBJ (PRO you))))
```

the parser omits the IP-SUB node:

```
(CP-QUE-MAT (WNP (WPRO What))
            (ADVP (ADV then))
            (VBP think)
            (NP-SBJ (PRO you)))
```

In this case, the parser's miss actually contains enough information for us to identify the output as an instance of `verb-sbj`. We therefore wrote 4 queries to retrieve systematic errors of this sort (missing vs. superfluous IP crossed with presence vs. absence of auxiliary *do*). Combining the results obtained in this way with the original results for questions tends to yield modest score improvements, as shown in the last two rows of Table 2 labeled `do-sbj+` and `verb-sbj+`.

7.2.2 Differences in Dev and Test Results

The results for the questions are significantly higher for the dev than for the test section. Since the dev section was used in training to determine the best model, as mentioned in Section 5.1, it might be thought that the results are naturally biased in favor of that section. But this idea is not consistent with the results in Table 1, where the evalb scores for the dev and test sections are quite similar.

A closer look at Table 2 suggests that the discrepancy between the dev and the test scores is an artifact of how the sentences types are distributed

[13] Future work could benefit from adapting the parser error analysis technique in Kummerfeld et al. (2012).

[14] It may be worth noting that parentheticals are encoded by a PRN function tag in PPCEME, whereas PTB encodes them by a separate PRN node. It may be worth investigating whether the PTB convention would improve accuracy in the cases at hand.

split	#	recall	prec	f1
dev				
0	43	67.44	90.62	77.33
1	36	94.44	94.44	94.44
2	51	74.51	97.44	84.44
3	49	67.35	91.67	77.65
4	22	95.45	91.30	93.33
5	3	66.67	100.00	80.00
6	84	89.29	98.68	93.75
7	99	92.93	98.92	95.83
test				
0	29	79.31	79.31	79.31
1	36	75.00	93.10	83.08
2	38	63.16	75.00	68.57
3	15	53.33	72.73	61.54
4	3	33.33	100.00	50.00
5	17	94.12	84.21	88.89
6	17	70.59	100.00	82.76
7	35	88.57	93.94	91.18

Table 3: Breakdown of the `verb-sbj` Scores for the Dev and Test Sections. The # of occurrences adds up to 387 for the dev section and 190 for the test section.

in the dev/test sections of each split. In contrast to the questions, the dev and test scores for negative declaratives are roughly equal. The numbers of tokens in each split are well-balanced across the two sections, and the standard deviations are low by comparison to the other sentence types. For the other queries, though, the number of gold structures is either very low (negative imperatives) or badly distributed across the dev/test sections (questions). In both cases, a better result correlates with a lower standard deviation rather than with a consistently better result on either the dev or the test section. For `do-not-imp`, the test section has a higher score (81.83) than the dev section (71.83), and its standard deviation, though still high (34.0), is lower than that for the dev (44.7). By contrast, for `verb-not-imp`, it is the dev section that has a higher score (88.22 vs. 82.24) with a lower standard deviation (4.2 vs. 14.9). This pattern is repeated for the questions (both `do-subj` and `verb-sbj`), where the dev sections have higher scores and lower standard deviations than do the test sections.

A more detailed look at the `verb-sbj` scores in the dev section sheds even further light on the matter. Table 3 breaks down the scores for the dev and test sections by split. The dev results benefit from the scores for splits 6 and 7, which are both relatively numerous and high-scoring. In particular, in split 7, the dev section contains excerpts from the New Testament (authnew-e2) and a transcript of a trial (oates-e3). In both of these source texts, questions occur at a higher rate than they do in other sources, and they tend to be simple questions without parentheticals (unlike those sort discussed in Section 7.2.1). In other words, split 7 contains many easy questions.

8 Conclusion

We have presented the first results on parsing PPCEME, defined an 8-fold cross-validation split, and evaluated the parser using a query-based measure connected to an overarching project in theoretical linguistics. The precision scores are generally very good, but we identified some of the problematic structures for recall and noted that even with the use of cross-validation for evaluation, the results are highly variable. In future work, we plan to use BERT embeddings and experiment with different parser models to improve parser accuracy, even though, as noted in Section 7.2, we are able to accept limits on parser accuracy for our purposes, as long as the parser's erorrs are unbiased. It is possible that parsers based on well-defined grammatical structures, such as Flickinger (2011) or (Kasai et al., 2018) will eliminate the nonsensical structures discussed in Section 7.2.1. Another alternative, in a different direction, is to use sentence embeddings derived from word embeddings, as in Arora et al. (2017), to identify the desired sentences directly, without using a parser at all.

At the same time, we recognize that the high variability revealed by our cross-validation procedure calls for evaluation on an extended set of diagnostic sentences - a task that we plan to tackle in three ways:

(1) We will extend query-based precision testing to Santorini (2021), a corpus of roughly 325,000 words of Early Modern English consisting entirely of diagnostic sentence types.

(2) We will further extend query-based testing to a representative sample of EEBO. Though we have no gold trees for EEBO, we can evaluate precision by manually checking the query hits found in the sample. This will also allow us to compare parser performance across EEBO and PPCEME. While we would expect roughly similar scores, it would not be surprising to find a decline in accuracy due in part to the tokenization approximations and OCR errors in EEBO mentioned in Section 4.

(3) In the latter case, we find ourselves in a position to give a quite rigorous quantitative estimate of the size of such a decline. As it turns out, about 40% of PPCEME overlaps roughly in underlying source

text with EEBO, and we have carried out a word alignment between the parallel texts. Thus, after training the parser on the non-overlapping 60% of PPCEME and running our queries on the parser output for both parallel texts, comparing the query-based results should give us the desired estimate for any performance dropoff to be expected when parsing the rest of EEBO.

Acknowledgments

This work was supported by NSF grant BCS 13-046668, ”Annotating and extracting detailed syntactic information from a 1.1-billion-word corpus.” We thank the three anonymous reviewers for their comments. We would also like to acknowledge here the tremendous debt, intellectual and in other ways, that this work owes to the late Prof. Anthony Kroch.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*.

Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. Scibert: Pretrained contextualized embeddings for scientific text. *arXiv preprint arXiv:1903.10676*.

Don Blaheta. 2003. *Function Tagging*. Ph.D. thesis, Brown University.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Aaron Ecay. 2015. *A multi-step analysis of the evolution of English do-support*. Ph.D. thesis, University of Pennsylvania.

Dan Flickinger. 2011. Accuracy vs. robustness in grammar engineering. In E. M. Bender and J. E. Arnold, editors, *Language from a cognitive perspective: Grammar, usage, and processing*, CSLI Publications, pages 31–50. Stanford.

Ryan Gabbard, Seth Kulick, and Mitchell Marcus. 2006. Fully parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 184–191, New York City, USA. Association for Computational Linguistics.

Charlotte Galves. 2020. Relaxed V-Second in Classical Portuguese. In Rebecca Woods and Sam Wolfe, editors, *Rethinking Verb-Second*, pages 368–395. Oxford University Press.

Charlotte Galves, Aroldo Leal de Andrade, and Pablo Faria. 2017. Tycho Brahe Parsed Corpus of Historical Portuguese. `http://www.tycho.iel.unicamp.br/~tycho/corpus/texts/psd.zip`.

Kyle Gorman and Steven Bedrick. 2019. We need to talk about standard splits. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2786–2791.

Qiao Jin, Bhuwan Dhingra, William W Cohen, and Xinghua Lu. 2019. Probing biomedical embeddings from language models. *arXiv preprint arXiv:1904.02181*.

Andres Karjus. 2020. *Competition, selection and communicative need in language change: An investigation using corpora, computational modelling and experimentation*. Ph.D. thesis, University of Edinburgh.

Jungo Kasai, Robert Frank, Pauli Xu, William Merrill, and Owen Rambow. 2018. End-to-end graph-based tag parsing with neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1181–1194.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Anthony Kroch. 1989. Reflexes of grammar in patterns of language change. *Language Variation and Change*, 1(3):199–244.

Anthony Kroch. 2020. Penn Parsed Corpora of Historical English. LDC2020T16 Web Download. Philadelphia: Linguistic Data Consortium. Contains Penn-Helsinki Parsed Corpus of Middle English, second edition, Penn-Helsinki Parsed Corpus of Early Modern English, and Penn Parsed Corpus of Modern British English.

Anthony Kroch and Beatrice Santorini. 2021. Penn-BFM Parsed Corpus of Historical French, version 1.0. https://github.com/beatrice57/mcvf-plus-ppchf.

Anthony Kroch, Beatrice Santorini, and Lauren Delfs. 2004. Penn-Helsinki Parsed Corpus of Early Modern English (PPCEME). CD-ROM, first edition, release 3. http://www.ling.upenn.edu/ppche/ppche-release-2016/PPCEME-RELEASE-3.

Anthony Kroch, Beatrice Santorini, and Ariel Diertani. 2016. Penn Parsed Corpus of Modern British English (ppcmbe2). CD-ROM, second edition, release 1. http://www.ling.upenn.edu/ppche/ppche-release-2016/PPCMBE2-RELEASE-1.

Anthony Kroch, Ann Taylor, and Donald Ringe. 2000a. The Middle English verb-second constraint: A case study in language contact and language change. In Susan Herring, Lene Schoessler, and Peter van Reenen, editors, *Textual parameters in older language*, pages 353–391. Benjamins.

Anthony Kroch, Ann Taylor, and Beatrice Santorini. 2000b. Penn-Helsinki Parsed Corpus of Middle English (PPCME2). CD-ROM, second edition, release 4. http://www.ling.upenn.edu/ppche/ppche-release-2016/PPCME2-RELEASE-4.

Seth Kulick, Anthony Kroch, and Beatrice Santorini. 2014. The Penn Parsed Corpus of Modern British English: First parsing results and analysis. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 662–667, Baltimore, Maryland. Association for Computational Linguistics.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*.

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

France Martineau, Paul Hirschbühler, Anthony Kroch, and Yves Charles Morin. 2021. MCVF Corpus, parsed, version 2.0. https://github.com/beatrice57/mcvf-plus-ppchf.

Paola Merlo and Gabriele Musillo. 2005. Accurate function parsing. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 620–627, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Taesun Moon and Jason Baldridge. 2007. Part-of-speech tagging for Middle English through alignment and projection of parallel diachronic texts. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 390–399, Prague, Czech Republic. Association for Computational Linguistics.

Sebastian Nagel. 2016. Cc-news. https://commoncrawl.org/2016/10/news-dataset-available/.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Beth Randall. 2010. CorpusSearch 2: a tool for linguistic research. Download site: http://corpussearch.sourceforge.net/CS.html. User guide: https://www.ling.upenn.edu/~beatrice/corpus-ling/CS-users-guide/index.html.

Beatrice Santorini. 2021. Parsed Ellegård 1953 dataset. https://github.com/beatrice57/ellegard-examples-parsed.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109, Dublin, Ireland. Dublin City University.

Satoshi Sekine and Michael Collins. 2008. evalb. http://nlp.cs.nyu.edu/evalb/.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistic (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Ann Taylor, Arja Nurmi, Anthony Warner, Susan Pintzuk, and Terttu Nevalainen. 2006. Parsed Corpus of Early English Correspondence. Distributed through the Oxford Text Archive.

Ann Taylor, Anthony Warner, Susan Pintzuk, and Frans Beths. 2003. York-Toronto-Helsinki Parsed Corpus of Old English Prose. Distributed by the Oxford Text archive.

Text Creation Partnership. 2019. Early English Books Online. https://textcreationpartnership.org/tcp-texts/eebo-tcp-early-english-books-online/. Version 2019-04-25.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Joel C. Wallenberg. 2016. Extraposition is disappearing. *Language*, 92(4):e237–e256.

Joel C. Wallenberg, Rachael Bailes, Christine Cuskley, and Anton Karl Ingason. 2021. Smooth signals and syntactic change. *Languages*, 6(2):60.

Joel C. Wallenberg, Anton Karl Ingason, Einar Freyr Sigurdhsson, and Eiríkur Rögnvaldsson. 2011. Icelandic Parsed Historical Corpus (IcePaHC), v0.9. http://www.linguist.is/icelandic_treebank.

Yi Yang and Jacob Eisenstein. 2016. Part-of-speech tagging for historical English. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1318–1328, San Diego, California. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Richard Zimmermann. 2017. *Formal and quantitative approaches to the study of syntactic change: Three case studies from the history of English*. Ph.D. thesis, University of Geneva.

A PPCEME Part-of-Speech Modifications

In addition to the changes described in the main text, we changed the tag MD0 to MD. MD0 is an untensed modal, as in he will can or to can do something. There are only 4 cases, as this is an option that had mostly died out by the time of Early Modern English.

There are also cases where words that are ordinarily spelled as a single orthographic token are sometimes split into several tokens. PPCEME represents the former case with a single POS tag and the latter as a constituent whose non-terminal is the POS tag, with the words given numbered segmented POS tags - for example, (ADJ alone) vs. (ADJ (ADJ21 a) (ADJ22 lone)). We modified all such tags by removing the numbers, and appending _NT to the nonterminals, in order to more clearly distinguish between POS tags and nonterminals. In this example, the resulting structure would be (ADJ_NT (ADJ a) (ADJ lone)).

B PPCEME Issues

B.1 Metadata

In addition to the changes described in Section 2.2.1, we removed the metadata under CODE, META, and REF nodes. In cases where CODE dominated a leaf, removing the leaf resulted in an ill-formed tree. The 267 trees in question were removed, as were 576 trees rooted in META (usually stage directions for a play) and 9 trees containing BREAK.

In addition, before carrying out the above modifications, we changed all instances of (CODE <paren>) and CODE <$$paren>) to (OPAREN -LRB-) and (CPAREN -RRB-), respectively. We did this in order to retain the parentheses that otherwise, being daughters of CODE, would have been deleted.

Our counts of number of words and sentences differ slightly from Yang and Eisenstein (2016). We aim to resolve these discrepancies, which are probably related to small differences of preparation of the type just discussed.

B.2 Function Tags

We exclude certain tags that occur very rarely in PPCEME (ADT, CLF, COM, ELAB, EXL, RFL, TAG, TMC, TPC, XXX, YYY). Table 4 shows the frequency for each of the remaining 31 tags in the entire corpus, for nonterminals with a non-empty yield. For convenience, the tags are organized into six groups. The syntactic and semantic groups are roughly similar to those groups for the PTB, as presented in Gabbard et al. (2006). The other groups include tags that differ significantly from those in the PTB, as noted in Section 2.2.1. For the full set of PPCEME function tags, see https://www.ling.upenn.edu/hist-corpora/annotation/labels.htm.

Tag	Description	Frequency
	Syntactic	**37.23**
SBJ	subject	21.00
OB1	direct object	11.96
OB2	indirect object	1.20
SPR	secondary predicate	0.28
MSR	measure	1.17
POS	possessive	0.86
VOC	vocative	0.77
	Semantic	**7.93**
DIR	directional	0.50
LOC	locative	0.84
TMP	temporal	3.09
ADV	adverbial	3.50
	CP only	**8.83**
CAR	clause-adjoined	0.55
REL	relative clause	3.36
THT	THAT clause	2.52
CMP	comparative	0.53
QUE	question	1.35
FRL	free relative	0.33
EOP	empty operator	0.19
	IP only	**9.67**
INF	infinitive	4.59
PPL	participial	2.18
IMP	imperative	1.12
SMC	small clause	0.90
PRP	purpose	0.46
ABS	absolute	0.42
	CP or IP	**33.10**
SUB	subordinate	14.52
MAT	matrix	12.66
SPE	direct speech	5.64
DEG	degree	0.28
	Miscellaneous	**3.23**
PRN	parenthetical	2.60
RSP	resumptive	0.33
LFD	left-dislocated	0.30

Table 4: Function Tags in PPCEME with Their Frequencies. The tags are organized into 6 groups, with combined frequency by group in boldface.

C Train/Dev/Test Split

Table 5 summarizes the composition of the train/dev/test sections across the cross-validation 8 splits; specifically, the total number of documents, the total number of tokens, and the percentage of total tokens in each section. Since the

section	# files		# tokens		% of split	
train	205.88	(13.34)	1743211.25	(10441.53)	89.65	(0.54)
dev	12.50	(7.15)	101000.12	(4081.82)	5.19	(0.21)
test	13.62	(7.91)	100268.62	(7832.66)	5.16	(0.40)
OVERALL	232	(0.00)	1944480	(0.00)	100	(0.00)

Table 5: Mean number of files and tokens for train/dev/test sections across the 8 cross-validation splits (standard deviations are presented in parentheses). The percentage of tokens in each section is also presented (in the **% of split** column).

partitioning process is performed at the level of PPCEME source files, and these files differ substantially in size, there is some variation in these numbers across the splits. For this reason, we report standard deviations as well as means. The final row ("OVERALL") gives numbers for a complete split (i.e., the train/dev/test sections combined); as these are constant across each split, the entries in this row have a standard deviation of zero. As can be seen, overall the splits attain the target 90-5-5 breakdown; e.g., the train section on average comprises 89.65% of the total tokens with a standard deviation of 0.54%.

As mentioned in the main text, the corpus consists of text from three main time periods (e1, e2, e3),[15] and we aimed to balance the time periods equally within each split, to the extent possible given that we treated the files as atomic units. Table 6 shows the breakdown by period. Similar to Table 5, mean/standard deviation for total number of documents/tokens are presented for each time period in each section. Additionally, for each time period, the table reports the mean percentage of each split (in tokens) from each time period. The marginals provide numbers combining across time periods (the "ALL PERIODS" row) and sections (the "ENTIRE SPLIT" column). For example, the training section contains on average 1,743,211.25 tokens, with on average 32.85% coming from time period e1, 36.61% from e2, and 30.53% from e3.

D ELMo Embeddings Trained on EEBO

In addition to the normalizations discussed in Section 4, we follow (Ecay, 2015) in removing information under `NOTE`, `SPEAKER`, and `GAP`, as well as `L` ("line of verse") which was not appropriate for our searches. In future work, we will likely revise this to keep the text but with some meta-tags to indicate its origin.

The extracted text underwent Unicode normalization to NFC form in order to eliminate spurious surface differences between tokens. The resulting text contained 642 unique characters, 381 of which occurred fewer than 200 times. Manual inspection of these uncommon characters revealed that while some of them made sense in context (e.g., within sections of Greek or Latin text), many seemed to be spurious characters due to OCR errors (e.g., WHITE RECTANGLE 0X25AD). Consequently, we elected to filter out all sentences containing characters occurring fewer than 200 times. This eliminated 4139 lines, with 9,341,966 remaining for training (consisting of 1,168,749,620 tokens).

The ELMo embeddings were trained using TensorFlow maintained and distributed by AllenNLP at `https://github.com/allenai/bilm-tf` using the default model configuration.

E Model and Evaluation

Table 7 shows the hyperparameter settings used in the Berkeley Neural Parser. These are all the default settings for these parameters. We added a parameter `max_epochs`, used to set the maximum number of epochs. For the cross-validation training reported, we set `max_epochs=50`.

F Function Tag Evaluation

Function tags are typically removed by evalb before it compares bracket labels, and we have not modified this. To evaluate function tag recovery, we follow the approach of Gabbard et al. (2006). who in turn follow Blaheta (2003). Under this approach, function tags are evaluated only for nonterminals that evalb counts as matches. For example, an NP-SBJ in the parsed tree corresponding to an NP-SBJ in the gold tree counts as a match for SBJ. But an NP-OB1 in the parsed tree corresponding to an NP-SBJ node in the gold tree (which is possible since

[15] For details regarding the PPCEME time periods (e1, e2 and e3) see `https://www.ling.upenn.edu/hist-corpora/PPCEME-RELEASE-3/description.html`

	train section			dev section			test section			ENTIRE SPLIT		
period	# files	# tokens	% train	# files	# tokens	% dev	# files	# tokens	% test	# files	# tokens	% split
e1	72.88 (6.51)	572672.62 (11974.31)	32.85 (0.79)	4.25 (3.01)	33178.50 (7078.50)	32.98 (7.36)	4.88 (4.55)	31369.88 (8193.65)	31.50 (8.67)	82 (0.00)	637221 (0.00)	32.77 (0.00)
e2	66.00 (4.38)	638269.88 (13490.18)	36.61 (0.60)	4.00 (2.51)	34844.62 (6382.81)	34.40 (5.41)	4.00 (2.14)	35186.50 (7767.44)	34.89 (6.18)	74 (0.00)	708301 (0.00)	36.43 (0.00)
e3	67.00 (5.18)	532268.75 (7066.41)	30.53 (0.35)	4.25 (3.96)	32977.00 (5211.71)	32.63 (4.65)	4.75 (3.45)	33712.25 (5592.81)	33.60 (4.70)	76 (0.00)	598958 (0.00)	30.80 (0.00)
ALL PERIODS	205.88 (13.34)	1743211.25 (10441.53)	100 (0.00)	12.50 (7.15)	101000.12 (4081.82)	100 (0.00)	13.62 (7.91)	100268.62 (7832.66)	100 (0.00)	232 (0.00)	1944480 (0.00)	100 (0.00)

Table 6: Mean number of files and tokens for train/dev/test sections within each of three time periods (e1, e2, and e3) across the 8 cross-validation splits. The **% train/dev/test** columns indicate the % of total train/dev/test tokens for each time period. Standard deviations are presented in parentheses.

hyperparameter	value
attention_dropout	0.2
batch_size	32
char_lstm_input_dropout	0.2
checks_per_epoch	4
clip_grad_norm	0.0
d_char_emb	64
d_ff	2048
d_kv	64
d_label_hidden	256
d_model	1024
d_tag_hidden	256
elmo_dropout	0.5
encoder_max_len	512
force_root_constituent	'auto'
learning_rate	5e-05
learning_rate_warmup_steps	160
max_consecutive_decays	3
max_len_dev	0
max_len_train	0
morpho_emb_dropout	0.2
num_heads	8
num_layers	8
predict_tags	True
relu_dropout	0.1
residual_dropout	0.2
step_decay_factor	0.5
step_decay_patience	5
tag_loss_scale	5.0
max_epochs	50

Table 7: Hyperparameters Used with the Berkeley Neural Parser.

	Function Tags
dev	94.90 ± 1.54
test	95.55 ± 0.87

Table 8: Cross-validation Function Tag Results.

the function tags do not count for evalb) counts as a recall error for SBJ and as a precision error for OB1.

Table 8 shows dev and test section scores for the function tags using this scoring method, analogously to Table 1. To explore these numbers in greater depth, Table 9 breaks down the function tag results for the first cross-validation split, organized as in Table 4. By far the most significant cause for a decreased score is the SPE tag indicating direct speech. Though one of the most common tags, with a frequency of 8.21%, it attains an f1 score of only 50.75. The discrepancy reflects the absence in PPCEME of consistent clues for direct speech (such as quotation marks) that are available, say, in modern newswire text subject to strict style guidelines. Fortunately, however, SPE is not highly relevant for the purposes at hand.

Tag	Description	Frequency	F1
Syntactic		37.03	96.57
SBJ	subject	21.57	98.06
OB1	direct object	11.63	95.33
MSR	measure	1.18	92.42
OB2	indirect object	1.06	92.10
POS	possessive	0.75	96.31
VOC	vocative	0.52	92.82
SPR	secondary predicate	0.32	76.07
Semantic		7.81	95.57
ADV	adverbial	4.38	97.03
TMP	temporal	2.43	93.44
DIR	directional	0.52	95.78
LOC	locative	0.48	93.19
CP only		8.18	91.86
REL	relative clause	3.04	92.03
THT	THAT clause	1.80	95.07
QUE	question	1.46	95.23
CAR	clause-adjoined	0.67	76.78
CMP	comparative	0.63	96.97
FRL	free relative	0.48	81.57
EOP	empty operator	0.11	88.61
IP only		9.98	95.72
INF	infinitive	4.92	98.24
PPL	participial	2.35	98.59
IMP	imperative	1.26	90.83
SMC	small clause	0.83	95.68
PRP	purpose	0.46	75.98
ABS	absolute	0.17	69.92
CP or IP		34.38	88.24
SUB	subordinate	14.94	98.72
MAT	matrix	10.92	98.07
SPE	direct speech	8.21	50.75
DEG	degree	0.31	86.31
Miscellaneous		2.62	81.45
PRN	parenthetical	1.77	87.99
RSP	resumptive	0.47	58.62
LFD	left-dislocated	0.37	73.65
Total		100.00	92.83

Table 9: Function Tag Results for the Dev Section.