Penn-Helsinki Parsed Corpus of Early Modern English: First Parsing Results and Analysis

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

The Penn-Helsinki Parsed Corpus of Early Modern English (PPCEME), a 1.7-million-word treebank that is an important resource for research in syntactic change, has several properties that present potential challenges for NLP technologies. We describe these key features of PPCEME that make it challenging for parsing, including a larger and more varied set of function tags than in the Penn Treebank, and present results for this corpus using a modified version of the Berkeley Neural Parser and the approach to function tag recovery of Gabbard et al. (2006). While this approach to function tag recovery gives reasonable results, it is in some ways inappropriate for span-based parsers. We also present further evidence of the importance of in-domain pretraining for contextualized word representations. The resulting parser will be used to parse Early English Books Online, a 1.5 billion word corpus whose utility for the study of syntactic change will be greatly increased with the addition of accurate parse trees.

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; Kroch and Santorini, 2021; Martineau et al., 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).

PPCEME is of interest for NLP researchers as well as linguists since it differs from the Penn Treebank (PTB) (Marcus et al., 1993) along several dimensions, and it is not obvious that parsing technologies mostly based on the PTB can transfer over successfully. While the text in the PTB is rigidly standardized, subject to editorial guidelines, the source text in PPCEME is more chaotic, with a lack of standardization of spelling and punctuation. The annotation style is similar to that of the PTB, but it has more finely-grained Part of Speech (POS) tags, some common and important structures (e.g., coordination) are annotated differently than in the PTB, and perhaps most significantly, it makes far greater use of function tags. There has however been relatively little work in the NLP community (Moon and Baldridge, 2007; Kulick et al., 2014; Yang and Eisenstein, 2016) using PPCEME or its sister corpora, the Penn Parsed Corpus of Middle English, 2nd edition (PPCME2) and the Penn Parsed Corpus of Modern British English (PPCMBE)[1].

In this paper, we present first results for both parsing accuracy and function tag recovery for PPCEME, pointing out some limitations for the latter. Our parser is a slightly modified version of the self-attentive neural constituency parser introduced by Kitaev and Klein (2018). While constituent-parsing technology has seen notable gains in recent years, to our knowledge there has no work on recovering function tags within this framework, which is crucial for the larger context of this work, discussed below in this introduction. We use an approach to function tag recovery used earlier with the PTB (Gabbard et al., 2006) and show that while it gives reasonable results, it is inefficient and inappropriate for contemporary span-based constituency parsers, especially as the number of function tags increases.

Many of our decisions are informed by the larger context of this work: parsing the Early English Books Online (EEBO) corpus (Text Creation Partnership, 2019). While PPCEME has been used for building models of syntactic change, its usefulness for this work is limited by the fact that various

[1] All three corpora are collected in Kroch (2020)

Findings of the Association for Computational Linguistics: NAACL 2022, pages 578 - 593

phenomena of interest are too sparse within it to support reliable statistical modeling of language change (e.g. distinguishing between competition (Kroch, 1989) and drift (Karjus, 2020)). In contrast, EEBO covers the same time period, 1475 to 1700, with 1.5 billion words of text – a difference of three orders of magnitude. However, EEBO's potential as a resource for linguistic research remains unrealized because it is not linguistically annotated and its size renders manual annotation infeasible.

Our goal therefore is the creation of a parsed version of EEBO consistent with the structures in PPCEME. This parsed version of EEBO will be used for linguistic research on language change by searching for structures of interest based on the phrase structure and POS tags. The parsing model trained on PPCEME will be used to parse EEBO so that it too can be used for queries for structures of linguistic interest, although at a much larger scale than previously possible. For more details, we refer to Kulick et al. (2022).

This broader concern impacts our design choices for the parser; for instance, our decision to work on constituency parsing rather than dependency parsing – linguistic queries depend on phrase structure trees. Likewise, since the function tags and POS tags are used by the linguistic queries, it critical that the parser be able to generate this information. It also informed the development of our tokenizer for EEBO. While in this paper we use standard metrics for parser evaluation, we also refer to Kulick et al. (2022) for a discussion of an alternative method of evaluation based on precision and recall for the specific queries.

2 Previous Work

The PPCEME is one of several corpora designed for historical linguistics research that share annotation styles and design decisions. Others include the second edition of the Penn-Helsinki Parsed Corpus of Middle English (Kroch et al., 2000b) (PPCME2) and the Penn Parsed Corpus of Modern British English (Kroch et al., 2016) (PPCMBE). While there has been much linguistics research based on these corpora, there is relatively little previous NLP work using them. Kulick et al. (2014) described parsing PPCMBE, while Moon and Baldridge (2007) and Yang and Eisenstein (2016) focused on POS-tagging, the former on PPCME2, and the latter on PPCEME and PPCMBE.

This previous work has discussed characteristics of these corpora that differ from that of the PTB and the potential impact on models trained on these corpora. A focus has been on how to transform the annotation to be closer to that of the PTB, such as by transforming the phrase structure (Kulick et al., 2014) or by mapping the PPCME2 POS tag set into that of PTB (Moon and Baldridge, 2007; Yang and Eisenstein, 2016).

3 Major Differences between PPCEME and PTB

As mentioned in the introduction, the PPCEME annotation style differs from that of PTB in certain respects. We describe here two of the most significant differences in more detail as well as how they influenced our preprocessing decisions. We also give examples in Section 3.3 of the spelling variations present in the Early Modern English data.

3.1 PPCEME Complex Part-of-Speech Tags

The PPCEME POS tag set is much larger (n=353) than that typically used in parsing work (e.g., for PTB n=36). Consequently, it requires trimming to a more computationally tractable size. Of the 353 tags, 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.

3.2 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

[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.

tags in this way, while distinguishing other constituent roles by means of structural differences (e.g., adjoining a relative clause but not a complement clause) 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 consider 31 function tags 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 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 this topic has received only limited attention. We are not aware of any work to recover the function tags in the historical corpora. As discussed in the introduction, our larger goal requires including them in the parsing model.

For the remainder of this paper, we refer to the version of PPCEME using these 31 function tags as "ftags-31". We also report results for a version using no function tags at all ("ftags-0")[4] and for a version using ten of the tags that were of particular importance for some of the linguistic searches ("ftags-10"). For a detailed description of the tags see Section 10.

3.3 Standardization

As mentioned in the introduction, the source texts of Early Modern English text (both in PPCEME and EEBO) can have spelling variations of a sort that are not present in corpora such as the PTB. Table 1 shows some examples of these variants in the source text.

standard	variants
sheriff	sheriffe, sherife, sherif, schereffe*
showed	shew'd, shewed*, shewd, show'd, shevved
servant*	seruant, seruaunt
strive	striue, stryue*
knocked	knockt, knock'd, knokyd*
asked	ask'd, askt, ask't, askyd*
devil	divel, devill, divell, diuell, deuill, diuel, deuil
against	agaynst, againste, agaynste

Table 1: Some of the variant spellings present in the Early Modern English source text. Forms marked with an asterisk are used in example sentences in this paper.

In addition, there is also some inconsistency of punctuation in the source material, which interacts with annotation decisions concerning sentence segmentation. A period POS tag marks sentence-final punctuation and a comma POS tag marks sentence-internal punctuation, though these do not always match the actual token receiving that tag.

For example, consider the following text for one sentence/tree in PPCEME:

> And when he cam ǀ knokyd at the gatys .
> To whome anone one of the gentylmannys seruauntys askyd who was there ǀ

The first ǀ is annotated with a comma POS tag, and the second ǀ is annotated with a period POS tag. The period receives a comma POS tag, since it is sentence-internal.[5]

4 Cross-validation Splits

Recently, concerns have been raised over the validity of inferences drawn from static train/dev/test splits of a corpus; 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 as 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.

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

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

[4] E.g. `NP-SBJ` becomes `NP`.

[5] In terms of the syntactic annotation, *when he cam* is a PP preceding the main verb *knokyd*, and *To whome...there* is a CP-relative clause.

70-year 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.[6] 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 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 B.

5 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), news text from Common Crawl (Nagel, 2016), and discussion forums (Radford et al., 2019) - pretraining on the novel domain yields significant improvements (Lee et al., 2020; 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 3, our current work involves exactly such a mismatch, and so we pretrained ELMo embeddings on EEBO.[7] We used the same model configuration as Peters et al. (2018) for 11 epochs[8]. Pretraining was performed using the TensorFlow implementation maintained by AllenNLP[9] using the default model configuration. We then integrated the resulting embeddings, which have 1,024 dimensions, into the parser model (see Section 6). Further details regarding the pretraining may be found in Appendix C.

6 Model and evaluation

6.1 Parser Architecture

We use the parsing 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 the beginning and ending positions of the span and l its label. Each tree is assigned a score $s(T)$, which is decomposed as a sum of per-span scores:

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

The per-span scores $s(i, j, l)$ are themselves 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 then produces a span score from an MLP classifier (Stern et al., 2017). The highest-scoring valid tree is 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).

We consider four approaches for producing the word embeddings that serve as input to the encoder:

bert-base Contextual word embeddings are computed using BERT (Devlin et al., 2019) by retaining the output of the last layer for the final subword unit of each token. We use the *bert-base* model distributed via the *transformers* Python package[10], which is pre-trained on 3.3 billion words of modern English.

elmo (orig) As above, but using ELMo (Peters et al., 2018). We use the ELMo-original model distributed by AllenNLP[11], which was pre-trained on 1 billion words of modern English.

elmo (eebo) ELMo embeddings pre-trained on Early Modern English from EEBO. Uses the same model architecture as *elmo (orig)*.

char-lstm Tokens are represented as the concatenation of the outputs of a bidirectional character-level LSTM for the final character of the token.

[6] 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.

[7] 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.

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

[9] https://github.com/allenai/bilm-tf

[10] https://github.com/huggingface/transformers

[11] https://allennlp.org/elmo

All four of these embeddings types are character or subword based, which is particularly helpful for the PPCEME corpus due to the spelling variations. The subword units for BERT however are derived from its modern English training material.

Our implementation is based on version 0.2.0 of the Berkeley Neural Parser[12] with modifications to allow the use of ELMo.[13]. We train each of the 8 models (one for each cross-validation split) for 50 epochs, using the evalb (Sekine and Collins, 2008) score on the dev section as our criterion for saving the best model. For additional details regarding training and hyperparameters, see Appendix D.

6.2 Function Tags

We adopt the approach of Gabbard et al. (2006) for function tag recovery, in which the parser preprocessing step simply does not delete the function tags, and so nonterminals such as `NP-SBJ` are treated as an atomic unit. Since the decision whether to delete is part of the preprocessing, this approach does not require modifying the parser. As mentioned in Section 3.2, our preprocessing code for PPCEME creates three versions of the files, ftags-0, ftags-10, and ftags-31.

6.3 Evaluation metrics

During training, the parser uses the standard evalb on the dev set to determine the best model so far. evalb compares matching brackets between the gold and system output trees. However, since our models also predict POS tags, and evalb removes punctuation based on POS tags, inconsistent sentence lengths can arise if the gold and parsed trees have differing POS tags, resulting in "Error" sentences in the evalb output. For the scores reported in Section 7, we therefore use the modified evalb supplied with the Berkeley parser, originating from Seddah et al. (2014), which does not delete any words, so the differences in POS tags do not affect the sentence length.

Function tags are typically removed by evalb for the comparison of bracket labels, and we have not modified this. To evaluate the function tag recovery, we use the metric in Gabbard et al. (2006) which in turn follows Blaheta (2003). This evaluation compares function tags only for nonterminals that are counted as matches for evalb, For example, if a NP-SBJ node in the gold tree matches with a NP-SBJ node in the parsed tree, it is a match for SBJ, while if a NP-SBJ node in the gold tree matches with a NP-OB1 node in the parsed tree (which can happen since the function tags do not count for evalb), it is a recall error for SBJ and a precision error for OB1.[14]

POS evaluation is the standard tag accuracy as output by evalb.

config	ftags-0	ftags-10	ftags-31
	dev		
elmo-e	90.92 (1.86)	90.95 (1.79)	90.89 (1.83)
bert	90.37 (1.95)	90.37 (1.95)	90.37 (1.98)
elmo-o	88.06 (2.23)	88.13 (2.25)	88.09 (2.22)
char	87.28 (2.34)	87.43 (2.33)	87.33 (2.33)
	test		
elmo-e	90.56 (0.66)	90.62 (0.71)	90.53 (0.69)
bert	89.81 (0.76)	89.83 (0.80)	89.86 (0.77)
elmo-o	87.31 (0.85)	87.50 (0.84)	87.38 (0.81)
char	86.29 (0.93)	86.50 (0.96)	86.45 (0.90)

Table 2: Cross-validated F1 for the parser on PPCEME using no function tags, 10 function tags, or 31 function tags. The standard deviation for F1 is presented in parentheses. Scores are obtained using the version of evalb that does not delete punctuation, for reasons discussed in Section 6.3, and do not consider function tags in the matching of brackets. *bert*: bert-base, *elmo-e*: ELMo pretrained on EEBO, *elmo-o*: original ELMo embeddings, *char*: character LSTM

7 Pretraining comparison experiments

Table 2 presents parsing results for the dev/test sections of the 8 cross-validation splits described in Section 4. The rows are the four embedding representations described in Section 6.1 and the columns are the F1 scores (evalb bracket scores, as discussed in Section 6.3) for each of the three versions (ftags-0, ftags-10, ftags-31) of the training and evaluation data. Each cell shows the mean and standard deviation for that embedding representation and function tag version over the 8 splits. As there is no great precision/recall imbalance, we relegate the corresponding precision and recall numbers to a more complete version of this table in Appendix E.

For each of the three versions of the training and evaluation data, the best F1 scores are obtained using ELMo (EEBO), followed by BERT, ELMo (orig), and, char-LSTM. Moreover, ELMo (EEBO) outperforms BERT (to varying degrees) on every single combination of split/function tags. In particular, for the version with 31 function tags, which

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

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

[14] We combined this function tag evaluation with our own Python reimplementation of evalb.

config	ftags-0	ftags-10	ftags-31
dev			
elmo-e	98.14 (0.72)	98.17 (0.72)	98.14 (0.69)
bert	97.75 (0.78)	97.73 (0.83)	97.71 (0.89)
elmo-o	96.87 (0.90)	96.93 (0.92)	96.91 (0.95)
char	97.17 (0.82)	97.25 (0.84)	97.23 (0.79)
test			
elmo-e	98.29 (0.36)	98.30 (0.36)	98.30 (0.36)
bert	97.95 (0.35)	97.94 (0.36)	97.97 (0.35)
elmo-o	97.09 (0.45)	97.14 (0.43)	97.16 (0.46)
char	97.23 (0.41)	97.30 (0.42)	97.28 (0.45)

Table 3: Cross-validated POS accuracy on PPCEME using no function tags, 10 function tags, or 31 function tags. Standard deviations are in parentheses. The rows and columns are analogous to those in Table 2.

will be the focus of further analysis in the following sections, ELMo (EEBO) outperforms BERT by 0.67% absolute (t=6.05, p<1e-3) on the test set.

The fact that BERT is consistently outperformed by ELMo (EEBO) is particularly interesting given that (a) BERT in general outperforms ELMo when trained on similar material, (b) BERT in this case was trained on three times as much material as ELMo, and (c) the BERT model parameters were fine-tuned, while the ELMo parameters were not. This result underlines the importance of pre-training on in-domain materials, a fact that has also been observed for NLP tasks in biomedical (Lee et al., 2020), financial (Araci, 2019), and legal (Chalkidis et al., 2020) domains.

Overall, we note that the scores are a few points lower than the current state of the art for PTB (96.38% F1 on the test section (Mrini et al., 2020)). 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, they find that it impacts the F1 score for PPCMBE by about 2.5% absolute, an effect we expect to carry over to PPCEME.

8 Part-of-speech Results

While the evalb scores are of greater importance, POS accuracy also matters as the linguistic queries we plan to run on the eventual parsed EEBO sometimes refer explicitly to POS tags. We therefore consider the POS results in detail. As is apparent from Table 3, POS accuracy is highest for ELMo (EEBO) for all combinations of function tags and dev/test set, followed by BERT, then ELMo (orig); e.g., when using 31 function tags, ELMo (EEBO) outperforms BERT by 0.33% absolute (t=5.68, p<1e-3) on the test set.

tag	frequency	f1
N	11.85 (0.70)	97.21 (0.84)
P	11.59 (0.45)	99.29 (0.19)
D	7.58 (0.94)	99.71 (0.10)
PRO	7.01 (1.04)	99.72 (0.17)
,	6.71 (0.93)	99.72 (0.14)
CONJ	5.20 (0.47)	99.55 (0.22)
.	4.42 (0.38)	99.69 (0.10)
ADJ	4.34 (0.45)	95.95 (0.85)
NS	3.48 (0.40)	97.31 (0.99)
ADV	3.05 (0.27)	97.11 (0.58)
NPR	2.98 (1.07)	93.05 (4.37)
VB	2.94 (0.33)	98.68 (0.55)
PRO$	2.57 (0.43)	99.60 (0.13)
VBD	1.95 (0.78)	97.56 (0.95)
MD	1.77 (0.25)	99.43 (0.59)
Q	1.75 (0.06)	99.00 (0.24)
VAN	1.71 (0.16)	96.31 (0.90)
VBP	1.65 (0.38)	96.65 (0.72)
BEP	1.62 (0.48)	99.56 (0.31)
TO	1.50 (0.12)	99.67 (0.09)
C	1.21 (0.16)	98.81 (0.53)
WPRO	0.95 (0.12)	99.36 (0.40)
BED	0.89 (0.22)	99.75 (0.22)
NUM	0.89 (0.48)	97.47 (1.15)
VAG	0.82 (0.13)	96.41 (0.65)
NEG	0.76 (0.13)	99.77 (0.12)
BE	0.68 (0.10)	99.46 (0.29)
VBN	0.60 (0.16)	96.38 (0.98)
HVP	0.50 (0.08)	99.25 (0.94)
ADVR	0.50 (0.07)	97.41 (0.68)
RP	0.49 (0.12)	96.12 (1.48)
FW	0.45 (0.24)	87.14 (10.44)
VBI	0.41 (0.09)	92.79 (2.58)
HVD	0.35 (0.13)	99.41 (1.12)
WADV	0.31 (0.04)	97.31 (1.22)

Table 4: Cross-validated per-tag F1 of the ELMo (EEBO) configuration for the 35 most frequent POS tags on the PPCEME test section (using the 31 function tag set). The frequency column indicates the mean relative frequency of each tag in the test set. Standard deviations are in parentheses.

As the distribution of POS tags is highly unbalanced, we also break down the 98.30% score in Table 3 for (elmo-e, ftags-31, test) and report F1 by tag for the 35 most frequent POS tags (i.e., tags with a frequency ≥0.30%) in the PPCEME test set (Table 4[15]). While there is some variation among the tags, F1 is consistently high (>95%) with the exception of the imperative (VBI), proper noun (NPR)[16], and foreign word (FW). The imperative

[15]For descriptions of PPCEME POS tags in the table, see https://www.ling.upenn.edu/hist-corpora/annotation/labels.htm.

[16]The NPR score of 93.05% is likely due to known annotation inconsistencies in the corpus between N (common noun) and NPR. This also impacts the N score, which at 97.21% does lower our overall POS score since it is the most common tag.

tag is of particular concern for us as we anticipate future users of the parsed EEBO will rely heavily on it for retrieval of imperative clauses (Kulick et al., 2022). We therefore provide two examples of parser confusion between VBI and VBP (present).

8.1 Example 1

```
(a) (IP-MAT (CONJ &)
            (NP-SBJ *con*)
            (VBP stryue)
            (IP-INF (TO to)
                    (VB get)
                    (NP-OB1 (D that)))))
(b) (IP-IMP (CONJ &)
            (VBI stryue)
            (IP-INF (TO to)
                    (VB get)
                    (NP-OB1 (D that)))))
```

In this first example, tree (a) is the gold tree, while (b) is the parser output, which has both the wrong POS tag for *stryue* (strive) and the wrong function tag for the parent IP. `*con*` in the gold tree is an elided subject under conjunction, and in this case refers to a subject in the previous tree. Generating the correct parse requires, as it would for a human, referring to the preceding text. In isolation, the parse with VBI is perfectly reasonable, although incorrect in context.

Another aspect of this error concerns the common NLP practice, which we follow, of removing empty categories from the parser training material (Appendix A.5). In this case, the removal of `*con*` has the consequence that the training data has ordinary (i.e. non-imperative) sentences without an overt subject and a seemingly arbitrary assignment of VBP or VBI to verbs.

This example therefore shows both the need to refer to text beyond the sentence level, and how the training data is corrupted by the removal of empty categories.

8.2 Example 2

```
(a) (IP-IMP (PP (P For)
                (NP (NS Mice)))
            (PP (ADV+P therefore))
            (VBI lay)
            (NP-OB1 (NP (N Poyson))
                    (CONJP or Oatmeal
                           mixt
                           with
                           pounded
                           glass)))
(b) (IP-MAT (CONJ for)
            (NP-SBJ (NS Mice))
            (PP (P therefore))
            (VBP lay)
            (NP-OB1 (NP (N Poyson))...))
```

This second example shows the reverse error, in which the gold POS tag is VBI for *lay*, while the parser makes it a VBP. The reason is that it incorrectly parses *for Mice*, and makes *Mice* the NP-SBJ laying the *Poyson*. Tagging *lay* as a VBP is consistent with that mistake.

9 Out-of-vocabulary span labels and resource inefficiency

As shown in Table 2, for each configuration (row) there is little effect on the F1 parsing scores among ftags-0, ftags-10, and ftags-31, a result that is consistent with Gabbard et al. (2006) for the PTB. However, including the function tags significantly affects the label vocabulary, as we discuss before going on to the analysis of the function tag performance in Section 10.

As discussed in Section 6.1, the parser uses a classifier to produce the most likely label for a span. The parser determines the label vocabulary at training time, by collecting all the possible nonterminal labels. There are two complications in this determination of the label vocabulary.

First, for unary branches, the parser combines the different nonterminals into a single label. For example, for the structure `(NP (CP ...))`, which is a NP with a single child, a CP, the label is NP::CP. At parse time, if this label is the most likely for a span, then it is split up again into a parent NP and child CP node.

Second, since function tags are integrated into the label vocabulary, they multiply out over the cases with unary branching. For example, with the structure `(NP (CP-FRL ...))`, for a NP with a free relative CP child, the value in the label vocabulary will be NP::CP-FRL. But there can also be instances `(NP-SBJ (CP-FRL ...))`, `(NP-OB1 (CP-FRL ...))`, resulting in the entries NP-SBJ::CP-FRL and NP-OB1::CP-FRL in the label vocabulary.

Table 5 depicts the impact of these two factors on the label vocabulary. Even without any function tags, the collapsing of unary branches causes the label vocabulary to increase from an average of 87.62 to an average of 334.25. With all 31 function tags, this increase is even more pronounced, with the label vocabulary increasing from 250.00 to 937.88.

The label vocabulary used by the model is the "collapsed" value for the train section. This increase of 687.88 resource requirements during training, and we found that the training time could increase

section	ftags-0	ftags-31
unary branches not collapsed		
train	87.62 (1.51)	250.00 (2.27)
test	46.12 (2.17)	143.38 (6.05)
both	45.25 (1.39)	140.75 (5.76)
unary branches collapsed		
train	334.25 (5.73)	937.88 (12.64)
test	117.50 (6.59)	299.88 (17.93)
both	111.12 (5.64)	278.62 (14.99)

Table 5: Span label vocabulary size along two dimensions: (1) no function tags vs. all 31 tags, and (2) labels are collapsed/not collapsed on unary branches. The "train" and "test" rows indicate the span label vocabulary sizes for those sections along those dimensions. The "both" row shows the number of labels that are common to the train and test section vocabularies. All sizes are mean and standard deviation over the 8 splits.

config	ftags-10	ftags-31
dev		
elmo-e	97.94 (0.51)	94.90 (1.54)
bert	97.79 (0.60)	94.40 (1.42)
elmo-o	97.13 (0.66)	93.56 (1.69)
char	97.01 (0.69)	93.44 (1.60)
test		
elmo-e	97.90 (0.28)	95.55 (0.87)
bert	97.72 (0.27)	94.95 (0.94)
elmo-o	96.97 (0.28)	94.24 (0.81)
char	96.88 (0.35)	94.22 (0.82)

Table 6: Function tag scores for the different configurations, showing the mean and standard deviation over the 8 splits. The function tag score uses a comparison of the function tags for brackets that count as a match for evalb. There is no function tag score for ftags-0 since there are no function tags in that version of the training and evaluation data.

as much as 25% between trains of ftags-0 and ftags-31.

Much of the increased resource requirements is not required, though. The "test" rows show the corresponding numbers for the eight test sections, and the "both" row shows the label values that are in both the train and test sections. Even without function tags, an average of only 111.12 of the labels in the training section occur in the test section, while the test section has an average of 6.38 labels that are not in the training section $(117.50 - 111.12)$. The inefficiency worsens considerably with more function tags. For ftags-31, an average of only 278.62 of the 937.88 labels in the training section occur in the test section, and the test section has an average of 21.26 labels that are not in the training section $(299.88 - 278.62)$.

If a label is in the test section but not in the label vocabulary determined by the train section, the parser will not correctly predict spans with that label. They are in effect out-of-vocabulary span labels. While the parser score is not damaged more, due to the infrequency of these OOV labels, this shows the limits of the current approach. We consider these current results to be a baseline for future improvement.

10 Function tag analysis

Table 6 shows the score for the function tags over the different configurations, with either 10 or 31 function tags. The overall score for the function tags drops drastically between ftags-10 and ftags-31. For example, on the test sections, with ELMo EEBO it falls from a mean of 97.90% with ftags-10 to a mean of 95.55% for ftags-31.

To explore these results in more detail, we pick one cell in the table – ELMo EEBO with 31 tags, for the test sections – and look at the individual scores for the 31 function tags that make up the 95.55% result, as shown in Table 7. The tags are organized into 6 groups for expository convenience. The "syntactic" and "semantic" tags are roughly similar to those groups for the PTB, as presented in Gabbard et al. (2006). The other groups include tags that are very different than in the PTB, as mentioned in Section 3.2, such as the tags restricted to CP nodes and indicating properties of the CP clause. The asterisks in Table 6 indicate the ten tags included in ftags-10. The third column shows the frequency of each tag or group of tags among all the function tags being scored in the test section, and the F1 column shows the score for the tag or group of tags.

Examining the results for the tags included in ftags-31 but not in ftags-10, the main cause for the drop in score is the SPE tag for direct speech, which is one of the most common tags, with an average 3.91% frequency, but with an average score of 46.08% F1.[17] The PPCEME lacks the consistent clues for direct speech (such as quotation marks) that would be available in newswire text adhering to modern punctuation guidelines.

For example, the sentence `and he shall be welcome` has an IP-MAT-SPE (matrix IP, direct speech) as the root of the entire tree in gold annotation, while in the parser output it is IP-MAT.

[17] The somewhat greater drop in score for the dev section splits, from 97.94% to 94.90% in Table 6, is likely because in the dev sections the SPE tag performs just as badly, but is more common (6.79% frequency).

tag	description	frequency	f1
Syntactic		39.69 (1.55)	97.41 (0.32)
*SBJ	subject	23.02 (1.09)	98.71 (0.14)
*ACC	accusative	12.42 (0.79)	96.34 (0.62)
*DTV	dative	1.48 (0.36)	92.96 (2.70)
*VOC	vocative	0.69 (0.42)	93.88 (3.27)
MSR	measure	1.13 (0.24)	92.75 (1.28)
POS	possessive	0.72 (0.23)	97.97 (0.91)
SPR	sec pred	0.24 (0.06)	76.33 (5.07)
Semantic		7.72 (0.44)	95.09 (0.73)
TMP	temporal	2.91 (0.42)	93.54 (1.32)
ADV	adverbial	3.64 (0.41)	97.25 (0.42)
LOC	locative	0.70 (0.11)	91.89 (1.84)
DIR	directional	0.48 (0.09)	92.75 (1.56)
CP only		8.67 (0.60)	94.70 (0.69)
REL	rel clause	3.41 (0.51)	95.30 (0.90)
THT	THAT clause	2.47 (0.27)	97.60 (0.63)
*QUE	question	1.24 (0.53)	95.97 (0.42)
CAR	clause-adj	0.59 (0.15)	79.98 (2.81)
CMP	comparative	0.57 (0.14)	95.80 (0.66)
FRL	free relative	0.29 (0.06)	80.72 (3.78)
EOP	empty op	0.09 (0.02)	94.26 (3.66)
IP only		9.28 (0.95)	96.21 (0.46)
*INF	infinitival	4.46 (0.39)	98.97 (0.29)
PPL	participial	2.01 (0.44)	98.44 (0.72)
*IMP	imperative	1.26 (0.25)	93.50 (1.59)
SMC	small clause	0.73 (0.16)	97.74 (0.58)
PRP	purpose	0.45 (0.07)	71.76 (3.58)
ABS	absolute	0.37 (0.19)	86.07 (3.61)
CP or IP		31.78 (2.69)	94.22 (2.59)
*MAT	matrix	12.93 (0.70)	98.58 (0.29)
*SUB	subordinate	14.64 (0.96)	98.97 (0.22)
SPE	direct speech	3.91 (3.58)	46.08 (17.76)
DEG	degree	0.30 (0.05)	88.62 (2.35)
null	Misc	2.85 (0.32)	86.57 (2.73)
*PRN	parenthetical	2.17 (0.21)	90.00 (2.10)
RSP	resumptive	0.37 (0.08)	70.33 (7.98)
LFD	left-disl	0.31 (0.07)	79.37 (4.87)
Total		100.00 (0.00)	95.55 (0.87)

Table 7: Function tag breakdown. The third column shows the frequency of the tag in the test section, while the fourth column shows its F1 score. The tags are organized into six groups and we also indicate the combined frequency and F1 score for the tags of each group.

The parser (or for that matter, a person) cannot know that the sentence is direct speech without looking at the wider context, which is a conversation spanning a few sentences. The underlying problem here is related to that of the first example of a POS error in Section 8. Future work to recover the direct speech information will need to examine discourse contexts beyond just single sentences in isolation, as the parser does now.

The other scores show some variance in accuracy, although some of the most common tags (SBJ, SUB, MAT) have the highest scores. Considering the tags as groups, there are some drops in scores among some of the tags for CP or IP constituents (e.g. purpose clauses). These require more analysis, although in general the question of whether the function tag recovery is satisfactory can only be answered with reference to the larger aim of this work (that is, by determining if the tags are accurate enough to ensure accurate results from subsequent linguistic searches on automatically parsed material).

11 Conclusion and Future Work

In this work we have presented the first parsing results on the PPCEME, with a focus on challenges posed by its extensive set of function tags. Adapting an earlier approach to function tag recovery works reasonably well, while we point out some challenges for future work concerning improvements in accuracy and resource efficiency.

We have also demonstrated the continued importance of in-domain pretraining, as the parser configuration using ELMo trained on in-domain EEBO outperforms the configuration using BERT trained on modern English. We performed cross-validation to ensure that the results were robust and not an accident of a particular split.

Future work will proceed along a number of lines. Regarding the function tags, an obvious approach is to integrate a separate function tag classifier into the parser separate from one for the bare nonterminals and to combine them to produce the full nonterminal labels. We will also evaluate the function tags by their utility for linguistic searches on automatically parsed material, as mentioned at the end of Section 10.

Acknowledgments

This work was supported by NSF grant BCS 13-046668. 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 Professor Anthony Kroch.

References

Dogu Araci. 2019. FinBERT: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

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*.

Shu Cai, David Chiang, and Yoav Goldberg. 2011. Language-independent parsing with empty elements. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 212–216, Portland, Oregon, USA. Association for Computational Linguistics.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Yufei Chen, Yuanyuan Zhao, Weiwei Sun, and Xiaojun Wan. 2018. Pre- and in-parsing models for neural empty category detection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2687–2696, Melbourne, Australia. 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 (NAACL-HLT)*, pages 4171–4186.

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

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.

Charlotte Galves. 2020. Relaxed Verb 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 (ACL)*, 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*.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 136–143, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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.

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 (ACL)*, pages 3499–3505.

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 (ACL)*, pages 2676–2686.

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.

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.

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 (ACL)*, pages 662–667.

Seth Kulick, Neville Ryant, and Beatrice Santorini. 2022. Parsing Early Modern English for linguistic search. In *Proceedings of the Society for Computation in Linguistics 2022*, pages 143–157, online. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

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. 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.

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742.

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. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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.

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 (ACL)*, pages 818–827.

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/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30.

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 (NAACL-HLT)*, pages 1318–1328.

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 (ICCV)*, pages 19–27.

A PPCEME preprocessing

A.1 Metadata

In addition to the changes described in Section 3.2, 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). This is probably related to small differences of preparation of the type just discussed.

A.2 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))`.

A.3 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.

A.4 Function Tags

We exclude certain tags that occur very rarely in PPCEME (CLF, COM, TMC, RFL, ADT, EXL, YYY, ELAB, XXX, TAG, and TPC). Table 8 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 3.2. For the full set of PPCEME function tags, see `https://www.ling.upenn.edu/hist-corpora/annotation/labels.htm`.

A.5 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 (Johnson, 2002; Cai et al., 2011; Chen et al., 2018).

B Cross-validation Splits

Table 9 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 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") depicts numbers for a complete split (i.e., the train/dev/test sections combined); as this is constant across each split, the entries of this

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 8: Relative frequencies of the 31 retained function tags in PPCEME. The tags are organized into 6 groups, with combined frequency by group in boldface.

row have a standard deviation of zero. As can be seen, overall the splits have kept to 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)[18] 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 10 shows the breakdown by period. Similar to Table 9, mean/standard deviation for total number of documents/tokens are presented for each time period in each section. Additionally, for each time period, it 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.

C ELMo Pretraining on EEBO

C.1 Text Extraction

EEBO's XML files contain a great deal of metadata and markup in addition to the source 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. Information under `NOTE`, `SPEAKER`, and `GAP` elements were eliminated, as was information under the `L` ("line of verse") element, which was considered irrelevant for the linguistic searches envisioned for the final resource[19]. We also adopting his handling of `GAP` tags, which indicate the locations of OCR errors, which consists of mapping OCR errors to word-internal bullet characters - e.g., `Eccl•siasticall`.

C.2 Normalization

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 these 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).[20]

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

[19] In future work, we will likely revise this to keep the text but with some meta-tags to indicate its origin.

[20] This token count differs from that cited for EEBO in the main body of the paper (1.5 billion) due to improvements in preprocessing over the course of the work.

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 9: 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).

	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 10: 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.

C.3 Tokenization

After normalizing the extracted text into 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.[21]

D Parser training

D.1 Parser hyperparameters

Table 11 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`$= 50$.

The total number of parameters in each model is presented in Table 12. As the label vocabulary size depends both on the function tag set used (e.g., ftags-0 vs ftags-31) and the split the model was trained on as, model sizes due differ subtly from run-to-run (on the order of a few thousand parameters).

D.2 Training times

Mean training times in hours for each configuration are presented in Table 13. Each training run was performed using a single NVIDIA GTX 1080 GPU.

E Pretraining comparison experiments

Table 14 expands Table 2 to include recall and precision in addition to F1.

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

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 11: Hyperparameters used with the Berkeley Neural Parser.

config	ftags-0	ftags-10	ftags-31
bert	136M (1.47K)	136M (2.24K)	136M (3.25K)
elmo-e	27M (1.47K)	27M (2.24K)	27M (3.25K)
elmo-o	27M (1.47K)	27M (2.24K)	27M (3.25K)
char	27M (1.47K)	27M (2.24K)	27M (3.25K)

Table 12: Mean model sizes for each configuration using no function tags, 10 function tags, or 31 function tags. Standard deviations are in parentheses.

config	ftags-0	ftags-10	ftags-31
bert	24.38 (0.52)	24.88 (0.83)	30.75 (1.04)
elmo-e	26.00 (0.93)	31.50 (1.20)	31.75 (1.67)
elmo-o	26.50 (1.07)	30.75 (1.16)	31.50 (1.31)
char	13.12 (0.64)	15.62 (0.74)	16.50 (0.76)

Table 13: Mean training times (in hours) for each configuration using no function tags, 10 function tags, or 31 function tags. Standard deviations are in parentheses.

config	ftags-0	ftags-10	ftags-31
dev			
elmo-e (f1)	90.92(1.86)	90.95(1.79)	90.89(1.83)
(rec)	90.57(1.96)	90.64(1.91)	90.49(1.93)
(prec)	91.27(1.76)	91.26(1.69)	91.30(1.74)
bert (f1)	90.37(1.95)	90.37(1.95)	90.37(1.98)
(rec)	90.09(2.05)	90.02(2.06)	90.00(2.08)
(prec)	90.65(1.84)	90.72(1.84)	90.74(1.87)
elmo-o (f1)	88.06(2.23)	88.13(2.25)	88.09(2.22)
(rec)	87.47(2.35)	87.51(2.45)	87.38(2.37)
(prec)	88.66(2.12)	88.76(2.06)	88.81(2.07)
char (f1)	87.28(2.34)	87.43(2.33)	87.33(2.33)
(rec)	86.80(2.45)	86.86(2.49)	86.70(2.46)
(prec)	87.76(2.24)	88.02(2.18)	87.98(2.21)
test			
elmo-e (f1)	90.56(0.66)	90.62(0.71)	90.53(0.69)
(rec)	90.23(0.73)	90.31(0.80)	90.13(0.76)
(prec)	90.88(0.60)	90.93(0.64)	90.92(0.62)
bert (f1)	89.81(0.76)	89.83(0.80)	89.86(0.77)
(rec)	89.51(0.85)	89.46(0.87)	89.44(0.86)
(prec)	90.11(0.67)	90.20(0.74)	90.29(0.68)
elmo-o (f1)	87.31(0.85)	87.50(0.84)	87.38(0.81)
(rec)	86.71(0.92)	86.92(0.92)	86.69(0.90)
(prec)	87.91(0.79)	88.10(0.80)	88.08(0.73)
char (f1)	86.29(0.93)	86.50(0.96)	86.45(0.90)
(rec)	85.83(1.11)	85.89(1.06)	85.79(1.05)
(prec)	86.77(0.76)	87.12(0.88)	87.12(0.78)

Table 14: Cross-validation mean and standard deviation F1, recall, and precision parsing scores for each of the three versions of PPCEME, with no function tags, 10 function tags, or 31 function tags. "bert" is bert-base, "elmo-e" is elmo trained on EEBO, "elmo-o" is elmo with the original embeddings, and "char" is char-lstm. The F1 scores are the same as in Table 2.