Annotation of *you*-pronouns in Early Modern English texts using machine learning techniques

A project by Maria Irena Szawerna for

Machine learning for statistical NLP: Advanced LT2326

University of Gothenburg

PROJECT BACKGROUND

While the literature on Shakespeare's language, including his use of second-person singular pronouns, is extensive, many of the studies are plagued by the same issue: the lack of well-annotated sources. Although the works of Shakespeare are a finite collection of texts, they are still many, and no unified effort has been made to annotate them – this is especially problematic since different researchers may want to prioritize different features in their annotations. Nevertheless, there are features that are more universally needed in such annotation, one of them being the number of *you* forms, as except for the reflexive forms, they bear no indication of that feature, and they are important to differentiate between for studies on the distribution of second-person singular pronouns. Busse (2002) points out this issue in his monograph on Shakespearean pronouns and notes that since he extrapolates his results based on a smaller, manually annotated corpus, it means that his results sourced from that are merely an approximation, and that had he had all of Shakespeare's works annotated, the outcome may have been different; and this researcher is not the only one struggling with this limitation (pp. 30, 40-42). I myself have had to manually annotate some of Shakespeare's texts, both for a project in a course that I took in the fall of 2019 (Shakespeare's Early Modern English) and the MA thesis that I wrote and defended in 2022 (*Get thee to a nunnery:* The use of 2nd person singular pronouns in William Shakespeare's *As You Like It* and *Hamlet*) at Heidelberg University. Seeing as this has been a struggle not only for me but for a number of other researchers, I could not help but wonder if there is a computational solution to this issue. Therefore, I decided to try to develop a machine learning-based method for annotating the grammatical number on *you* pronouns as a project for the Machine learning for statistical NLP: Advanced course, with a special focus on utilizing pre-existing word embeddings and LSTMs for sentence-level analysis.

The issue of the annotation of historical linguistic data is also not an entirely new topic within the field of Computational Linguistics. Inquiries have been made into developing or adapting part of speech taggers for such data. Although none of them directly correspond to what this project's aim is, they can yield insights into the current trends in methods, as well as difficulties that can be encountered both when POS-tagging and feature-tagging. Hupkes and Bod (2016) evaluated how POS taggers developed on modern Dutch perform on historical data, and what data processing methods can improve the performance. They found that modern taggers do not perform well, but that modernizing the spelling

of the historical data can lead to significant improvements. They also suggest that using parallel corpora or transforming historical word forms to modern lemmas could further improve the results. Ultimately, the best performance, according to them, can be achieved by training a tagger on annotated historical data. Scheible, Whitt, Durell, and Bennet (2011) conducted a similar inquiry, but into historical German. They also concluded that while "off-the-shelf" taggers do not work very well with historical data, normalizing such data can result in significant performance improvements. Bollman (2013), also researching historical German, suggests that using hand-normalized historical data as training data, and machine-normalizing the rest of it also yields considerably better results; in addition, he suggests that this is a solution that is suitable for languages with little data available, as it was enough for him to manually normalize 250 tokens for the results to have improved significantly. Adesam and Bouma (2016) investigate methods for annotating historical Swedish, settling on a combination of spelling normalization, dictionary-sourced lemmata, and hand-annotated data. Their results show that not much hand-annotated data is needed for developing a decent POS tagger, but that that number also depends on the specifics of the annotation. Finally, a number of papers tackle historical English as well. Rayson, Archer, Baron, Culpeper, and Smith, (2007) are the authors of the perhaps seminal paper on POS-tagging Early Modern English, interestingly also basing their inquiry on Shakespeare's works. They conclude that unstandardized spelling has a major influence on the performance of existing taggers on historical data, and how normalizing it can alleviate that negative effect. The same conclusions are drawn by Hiltunen and Tyrkkö (2013) but are based on a corpus of medical texts from the same period. Finally, two papers discuss more modern machine-learning methods with respect to historical POS tagging. Yang and Eisenstein (2016) argue in favor of utilizing Feature Embeddings as a learnable element in a system, with normalization and domain adaptation in addition to that further improving the performance. In turn, Kulick, Ryant, and Santorini (2022) argue in favor of transformer-based word embeddings, like the one from BERT, being a promising method in the POS annotation of historical texts. Overall, the tendency is to try to test and adapt solutions developed on modern language data, either by tweaking the solutions itself or performing various normalization procedures on the historical data. The main issues when it comes to the data itself are the variable spelling and out-of-vocabulary tokens. While these normalization issues seem to also be relevant for grammatical number tagging, the performance results and methods are not directly comparable between the papers and this project, as most of the research discussed above focuses on POS tagging, not necessarily feature tagging.

DATA RESOURCES

Since this project is inspired by the research into Shakespeare's English, the training data naturally also had to be sourced from his writings. I already had hand-annotated data from my MA project, which I decided to reuse in this case. The data consisted of two plays, *As You Like It* and *Hamlet,* with all the *you*-pronouns annotated with _SG, _PL, or _UNK; the last tag was very rare and only used in cases where there was no contextual evidence to determine the number, or it was conflicting. The

texts were also stripped of the names of characters producing the utterances and other non-utterance text, a process which I called “trimming” of the plays. The original texts were sourced from The Folger Shakespeare, an online source that provides the texts of Shakespeare’s plays in a variety of formats; all of those plays’ texts are rendered in modernized spelling, though (The Folger Shakespeare). The annotated texts are available in the project repository.

METHODS

The annotated data was loaded into a Jupyter Notebook, where the rest of the project was conducted. First, the texts were tokenized into sentences using NLTK’s sent_tokenize(), and then the sentences containing one of the number tags were selected. They were also split into sentences containing only one such tag, and containing multiples, as in the beginning it was not clear if the project could handle more than one taggable pronoun per sentence. Subsequently, samples for training and testing were constructed. This was done by first creating a list of the tags present in the sentence (in case there was more than one), and then iterating over the sentence as tokenized using one of the BERT tokenizers to find the indices of the pronouns corresponding to the tags. Thus, if there was more than one *you*-pronoun in a sentence, more than one sample was created, since samples consist of the sentence, the class (which tag it is), and the index of the word that the tag/class corresponds to. The reasoning behind such a structuring of the samples was that while the word itself does not bear any markings of the number it refers to, the contextual information in the sentence may very well do that, as it seems that this is how the annotators are able to determine the number.

In the next stage, the training/testing split was defined, together with only singular and plural samples being selected. In that process, equal numbers of singular and plural samples were selected, and special attention was put to neither of the classes being overrepresented. Initially, the project was envisioned as a multi-class classification problem (with the three classes of singular, plural, and unknown), but the model was performing extremely poorly due to the unbalanced class representation. Since UNK was such a rare tag, it was discarded. Even then, without ensuring that both the training and testing sets are composed of 50/50 SG and PL, the results were not good. Even though ensuring this means that a certain number of already few samples must be discarded, it still improved the system’s performance, hence the final decision to construct the datasets like this. The samples were also shuffled within their splits. In the next section, data encoding and batching took place. Using PyTorch’s Dataset would not add anything to what was already available, so instead the dataloader was developed straight away, using PyTorch’s DataLoader. Batch size and shuffle could be defined independently and dropping the last batch was always set to true due to some peculiarities of linear layers. Because of that, also, batches were kept rather small, so that discarding the last one would not mean removing a significant chunk of the already sparse data. As for the collate function, a custom one was designed, where for each batch classes were encoded as 1 (singular) or 0 (plural); this was one-hot encoded when the problem was first envisioned as featuring more than two classes, but, in this case, we only need one probability

(of it, say, being singular – and by definition then if it is not, then it must belong to the other class). The sentences were tokenized using BERT tokenizer and fed to BERT; from the output from that model, the embeddings from the penultimate layer were sourced to serve as word representations. Finally, those two, together with the indices for the pronouns that were to be determined, were stacked and returned to the dataloader.

Subsequently, the model and its training loop, as well as a way to save a trained model, were defined. The ShakespeareanClassifier class, based on PyTorch's nn.Module, with a user-defined hidden layer size, consisted of a bidirectional LSTM layer that would process the sentence, and a classification layer that would process the output of the LSTM using 0.05 Dropout, two Linear layers separated by a LeakyReLU and concluding in a Sigmoid so that the output would represent a system-predicted probability. Dropout seems especially significant in this case since the source data is so sparse and this way at least some variety is introduced to it. From the LSTM output, the timestep representation at the index on the pronoun (this includes both directions of the LSTM) was taken to be fed to the classification layer. This meant that for one sentence more than one unique representation could be elicited, so for sentences with more than one *you*-pronoun there could be more than one outcome; additionally, the assumption was that taking them at the index of the pronoun would best represent the system's "understanding" of the sentence at the point when it gets to the word it has to make a prediction for. Within the training loop, for each epoch defined in the hyperparameters, a new dataloader was constructed from the training data. The sentences and indices from the batch were then fed to the model for every batch, and then compared to the true classes using Binary Cross Entropy Loss. Finally, optimization was carried out using Adam. Every five batches and at the end of every epoch the average loss was printed out. Finally, two small functions for saving and loading in the model using the pickle module were defined.

Then, the model was tested and evaluated. A testing loop was defined, where the testing data would be fed to the model, and the outputs and the true classes would be collected for all the data. Those could then be entered into another function, which nicely printed out measures as calculated using sklearn.metrics' accuracy, precision, recall, and F1 functions. Finally, a function creating a Pandas DataFrame containing the sentences, the predictions, and the true classes was defined, so as to allow for more qualitative analysis. In the final section, a large function was defined that would take a trained model and associated elements (hyperparameters, tokenizer) and an unannotated .txt file of a play from The Folger Shakespeare, and annotate it using the outputs from the model, as well as a small function that would save that output to a new .txt file. While the formatting of the annotated text is sometimes different from the input text (in terms of where newlines were inserted), it retains all of the content well enough, and, most importantly, features the annotations as predicted by the model.

RESULTS

All of these elements were then put into practice, with the *you*-sentences being extracted and turned into samples, those samples being split and saved, hyperparameters (learning rate of 0.00005, batch size of 8, hidden size of 1024, 10 epochs) being defined, and the model being initialized and then trained on the training data. These hyperparameters were tweaked as the model was re-run multiple times to see which ones work best, and the best performing model, with the staggering 80% accuracy, 76.9% recall, and 81% precision was saved (the exact numbers for these measures can be seen in the table below, including f1 as well). The same model and hyperparameters were then used for the annotation of the full text of *Macbeth*, which was then also saved. A qualitative analysis of the results in the DataFrame and in the text of the newly annotated play was carried out, where it was noticed that the sentences that were misclassified were most often short. The specific results in the DataFrame and the text of *Macbeth* can be found in the "project-acc80-with-anns.ipynb" notebook in the associated repository.

Table 1. The measures and results for the best performing model.

Measure	**Result**
Accuracy	0.8
Recall	0.7692307692307693
Precision	0.8108108108108109
F1	0.7894736842105263

DISCUSSION

What could be concluded from the results was that the model was relatively good at not over- or under-assigning one of the classes and that it did not perform badly at all, although it still struggled with some of the sentences. During the qualitative analysis it was revealed that the problematic sentences were those that were short, and, as a consequence, did not have much contextual information for the model to go off of. In the table below examples of sentences from testing data and *Macbeth* as well as their predicted and "golden standard" tags can be found (although for *Macbeth* there is no golden standard, but the selected sentences and their wider context should be enough to conclude whether the tag was assigned correctly or not). More of the results are of course available in the Jupyter Notebook file containing the code for the project.

When comparing these results with those featured in prior research, a conclusion can be made that the model did rather well, even though it was based on embeddings from a transformer model trained on modern English. Naturally, it was not as good as some of the results in the cited papers, but its goal was different, and there are ways to further improve it. One of the reasons for that is that the annotated

data already had modernized spelling, as such are the versions sourced from The Folger Shakespeare, and as mentioned in the discussion of the background literature, variation in spelling is a big factor in the performance of taggers. The other issue though, out-of-vocabulary tokens, was not addressed in this case. It is not clear how many words were mis-parsed by BERT's tokenizer, but during the development, it was noted that it was not unheard of for some of the words to be parsed in an incorrect way. Nevertheless, it does not seem like those tokens are the main issue that causes the suboptimal performance of the model. As can be noted from the qualitative analysis and in the table above, it was the short sentences with little contextual data that the model struggled with. It is likely that including more contextual data (either a fixed length chunk of text before and after the pronoun or the utterances before and after the one with the pronoun, as suggested during the discussion of the presentation of this project in class) would at least partially resolve this problem. It is also possible that excluding some of the non-utterance contextual information was not the right choice, as even when manually annotating, the names of speakers or characters present in the scene or information on who enters or exits the scene can yield valuable insights into whether a pronoun is singular or plural, as can be seen in the last sentence in the table, for example.

Table 2. Examples of correctly and incorrectly annotated sentences in the testing data and Macbeth.

Source	**Sentence**	**Prediction**	**Golden Standard**
Testing data	Beggar that I am, I am even poor in thanks; but I thank you, and sure, dear friends, my thanks are too dear a halfpenny.	PL	PL
	To you I give myself, for I am yours.	PL	SG
Macbeth	Kind gentlemen, your_PL pains Are registered where every day I turn The leaf to read them.	PL	PL
	[Enter Messenger.] What is your_PL tidings?	PL	SG

CONCLUSIONS

Overall, annotating for features that can be deduced from the context of a word using word embeddings and LSTMs seems promising, and as long as the bulk of the vocabulary can be tokenized and represented as word embeddings from a large language model, an architecture similar to the one presented in the model can work. More thought and experimentation need to be put into determining how much and what kind of context to provide for the words that need classifying, as clearly confining it to a sentence-level context is not sufficient for short sentences, and, in some cases, even out-of-utterance information may be indicative. It would be undoubtedly interesting how far this architecture

can be improved and what other features it can perform well for, as well as to what extent it can really be used on new data, e.g. whether it would perform well on other Early Modern English texts with modernized spelling, such as other playwright's works, but also prose, letters, and depositions or trial transcripts, as those either represent different writing styles or entirely different genres. Such a comparison could also yield insight into what the model is learning to draw conclusions about the number of the pronoun form.

Even if minor changes to the model's architecture and more substantial ones to how the relevant context is determined result in better performance, it is likely that the annotation, in some cases, will have to be double-checked by someone, but this kind of a model could possibly make the process of annotating data easier, and open the possibilities for machine-annotating (or at least pre-annotating) also for other features. It could also be claimed that perhaps unlike POS tagging, tagging for just one feature on one set of word forms is not extremely useful. However, as can be deduced from the sheer amount of literature on Early Modern English pronouns, in some cases such issue-specific annotation may be very useful, even if just purely for academic purposes.

REFERENCES:

Adesam, Y., & Bouma, G. (2016). Old Swedish Part-of-Speech Tagging between Variation and External Knowledge. *Proceedings of the 10th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, 32–42.

Bollmann, M. (2013). POS Tagging for Historical Texts with Sparse Training Data. *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 11–18.

Busse, U. (2002). *Linguistic Variation in the Shakespeare Corpus: Morpho-syntactic variability of second person pronouns.* John Benjamins Publishing Company.

Hiltunen, T., & Tyrkkö, J. (2013). Tagging Early Modern English Medical Texts (1500-1700). *Corpus Analysis with Noise in the Signal 2013* (conference).

Hupkes, D., & Bod, R. (2016). POS-tagging of Historical Dutch. *Proceedings of the Tentch International Conference on Language Resources and Evaluation (LREC'16),* 77-82.

Kulick, S., Ryant, N., & Santorini, B. (2022). Parsing Early Modern English for Linguistic Research. *Proceedings of the Society for Computation in Linguistics 2022*, 143–157.

Rayson, P., Archer, D., Baron, A., Culpeper, J., & Smith, N. (2007). Tagging the Bard: Evaluating the Accuracy of a Modern POS Tagger on Early Modern English Corpora. *Proceedings of Corpus Linguistics 2007*.

Scheible, S., Whitt, R. J., Durrell, M., & Bennett, P. (2011). Evaluating an 'off-the-shelf' POS-tagger on Early Modern German text. *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, 19–23.

The Folger Shakespeare. N.d. *Download Shakespeare's Plays, Sonnets, and Poems.* Available from: https://shakespeare.folger.edu/download/

Yang, Y., & Eisenstein, J. (2016). Part-of-Speech Tagging for Historical English. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1318–1328.

REPOSITORY:

The project repository can be found under the following link: https://github.com/Turtilla/ML2_project