Ancient VOC missives explored by new data mining techniques

S. Frinking, Q.C. Liem, L.G. Meima, M. Verma, and E.E. Zegelaar

Supervision: S.I. Arnoult, Vrije Universiteit Amsterdam

March 2020

Abstract

Optical character recognition (OCR) has made a considerable body of historical textual data accessible to be explored by natural language processing (NLP) techniques. However, not much research has been invested in the application of NLP technologies on historical language. In this paper we explore this field by testing and comparing two named entity recognition and classification (NERC) models on historical data of the Verenigde Oost-Indische Compagnie (VOC) era. The first is a pre-trained NERC model trained on contemporary data sets. The application of the pre-trained NERC systems on historical Dutch data sets gives very low evaluation scores. The second NERC model, which features a CNN-BiLSTM-CRF architecture, was trained on historical Dutch data from the VOC era. The second model displays significantly higher evaluation scores than the first model. As a result, we aimed at creating an application that makes use of the output from our era-specific NERC model and generates a timeline of people, locations, and traded goods.

keywords: historical Dutch, VOC, missives, NLP, NERC system, word embeddings, character embeddings, CNN, LSTM, CRF, timeline

1 Introduction

Thousands of Dutch missives from the era of the Verenigde Oost-Indische Compagnie (VOC) are ready to be explored. So far, little research is done on using natural language processing (NLP) techniques for analyzing historical texts. State-of-the-art models tend to be trained on contemporary Dutch but are not corrected for (early) modern Dutch. Nevertheless, NLP techniques could give us more and new insights in the history of the Dutch VOC.

In this paper we explore the VOC data by creating a timeline which places goods, ships and persons at the right place in time by using NLP techniques. In order to do this we need to automatically detect and classify named entities. Therefore, we will present two different models and compare these by using the standard metrics: precision, recall and F1-score. The first is a named entity recognition and classification (NERC) model trained on contemporary Dutch and applied on historical Dutch texts. The second is a NERC model that we have trained using VOC data. We manually annotated 27 000 tokens of different periods spreading over the 17th and 18th century. In the appendix more information is provided about the annotations.

We used two corpora for this project. The first is “Pieter van Dam’s Beschryvinge van de Oostindische Compagnie” [1] which dates from the 17th century. The second is the “Generale Missiven van Gouverneurs-Generaal en Raden aan Heren XVII der Verenigde Oostindische Compagnie”[2] which belongs to the period 1610-1761. Both corpora are from the period before the standardization of Dutch. It was in 1804 when the Dutch orthography and grammar was officially established by Siegenbeek[3]. This means there is a lot of variation in the spelling between the missives and within the spelling methods of the same author. For the first model, we could not correct for these inconsistencies. For the second model this was possible through the use of a Convolutional Neural Network (CNN) layer.

Low-resource named entity recognition continues to be a problem in NLP [4]. Although the corpora are large, the data is not annotated thus, it belongs to the low resource domain. Research in this field is mostly done on cross-lingual data in order to solve the problem of the lack of annotated resources. Xie et al. (2018) proposed two methods of lexical mapping via translation of words in a shared embedding space [5]. They used a bi-directional LSTM for both the character level and word level neural network, and we did so too. However, this model is built on modern languages. It is uncertain whether this cross-lingual approach can be used for the historical data. However, pre-trained models on modern data are used in a research for historical German [6]. They used unlabeled data trained on contemporary texts and created a synthetic masked language model (SMLM) based on a CRF, achieving good results. For Dutch historical data, current research is focused on modernising the spelling before applying modern NLP methods [7].

We propose three important questions that historians should pose in their research: (1) What happened? (2) Where did it happen? (3) Who were involved?. In the following sections we explain how we created an application (app) which employs a pipeline of NLP techniques to provide answers to these questions. However, considering the broad scope of these questions, we reformulated the *What*, the *Where*, and the *Who* to domain-specific content of our VOC corpus. The app consists of three components:

1. **What**: What were the most traded goods and how did this change over time?
2. **Where**: What locations were most prominent over time?
3. **Who**: Who were the most prominent people over time?

2 Model 1: Existing NERC systems on VOC data

First, we tested whether if it is possible to apply a NERC model trained on modern Dutch to historical texts and obtain qualitatively valid results.

As Table 1 illustrates, the evaluation scores of the BiLSTM-CRF applied to the VOC data are significantly low. To run this first model, we adapted our code from the CLTL text mining labs [8]. The errors mostly occur on words in which spelling method has changed over time. The system could not recognize these words, e.g. ’aanreekeningen’ (aanrekeningen (charges))[1], ’verwagting’ (verwachting (expectation)), ’expresselijk’ (expres (on purpose)). Errors also occur on words which do exist in the contemporary Dutch vocabulary. Furthermore, a lot of words which don’t have a capital letter are labeled as person or location. This is unexpected because most person and location names tend to be capitalized. Moreover, evident person names are sometimes labeled as O, such as ’Cornelis Hasselaar’ and ’Ewout van Dishoeck’.

[1] between brackets: first the contemporary spelling of the word, second the English translation

	precision	recall	F1	support
LOC	0.17	0.02	0.04	43
ORG	0.00	0.00	0.00	15
PER	0.00	0.00	0.00	37
micro avg	0.02	0.01	0.01	95
macro avg	0.08	0.01	0.02	95

Table 1: Evaluation Scores

3 Model 2: NERC system trained on VOC data

There are three main reasons why we opted to train our own custom model. Firstly, as the model trained on modern Dutch yielded dissatisfactory results, we decided that we want to create a model that is specifically trained on historical Dutch texts, profiting from domain-specific embeddings.

Secondly, it enables us to be more specific in the NERC labels used, because some labels are important for exploring VOC data which are not included in the current NERC models, such as 'goods' and 'ships'. Thirdly, we had to deal with a lot of spelling inconsistencies found in the VOC data. Incorporating a CNN component enables the model to generalize to a wider variety of spelling variations.

3.1 Gold data

3.1.1 Human Annotation

We collectively annotated over 27.000 tokens extending over 812 sentences. Named entities were annotated in IOB-format with the following labels: {DATE, LOC, PER, ORG, SHIP, GOODS, QUANTITY, MONEY}. Our main priority was generating sufficient gold data to train our model. Therefore, due to time and resource constraints there was no overlap in annotations between annotators, hence it was impossible to of calculate inter-annotator agreement. In Appendix A you can find the specific guidelines which we followed in annotation.

Furthermore, creating our own annotations enabled us to use our own domain-specific labels, which are not found in commonly used CoNLL corpora: GOODS, QUANTITIES and SHIPS. It should be noted that there is an unbalanced label distribution in our annotated corpus. This should be taken into account when evaluating the model's performance, and subsequently all results further down the pipeline.

The label distribution is as follows: {O: 26160, B-LOC: 674, I-PER: 448, I-DATE: 376, B-PER: 307, I-MONEY: 248, B-DATE: 241, B-GOODS: 238, B-QUANTITY: 206, B-MONEY: 187, I-QUANTITY: 186, B-SHIP: 146, I-LOC: 146, I-SHIP: 131, B-ORG: 98, I-GOODS: 68, I-ORG: 61}.

To make our model more robust, we attempted to supplement our training data with silver data. Each missive was accompanied by an index containing 'location', 'person', 'ship', and 'other' entities. We used this to extract entities from the text. We used a tokenizer to split the text in sentences an look up the indices in the text. Of course, not all spelling varieties of these indices were included. Due to this, we used fuzzy matching with a maximum Levenshtein distance of 1. This enabled us to find the indices in the text even if words were spelled differently. Through the use of fuzzy search, we matched all the indices in the text by considering one-letter substitutions of the words in text or the indices. When a close match is found in the text we copied the label of the indices to the string in the text.

However, the entities extracted by the procedure outlined above did not yield qualitatively results. Therefore, we excluded the silver data from our training data; opting for higher quality rather than quantity of training data.

3.2 Embeddings

Word embeddings enable us to capture semantic relations between words, based on the premise that semantically similar words occur in similar contexts. Since embeddings are trained with unsupervised learning, they can only learn from statistical patterns found in the text itself. And as our use of language is influenced by our culture, the embeddings can reflect the zeitgeist and period-specific culture. For example, Garg et al. (2018) and Zhao et al. (2019) reveal how word embeddings encode (covert) racial and gender biases through our use of language.[9, 10] Similarly, embeddings trained using the procedure described below, illustrate that in the VOC time period slaves were deemed more similar to livestock than to people (Table 2). Therefore, to adequately capture the nuances of the VOC time period, it became clear to us that we cannot extrapolate modern Dutch embeddings to historical texts.

	word	cosine similarity score
1.	lijfeygenen	0.794
2.	paarden	0.700
3.	lijffeygenen	0.689
4.	beesten	0.686
5.	familiën	0.680
6.	paerden	0.678
7.	vrouwen	0.666
8.	buffels	0.665

Table 2: Most similar words to 'slaven' *slaves* based on word embedding cosine similarity scores.

We trained embeddings on two main corpora: *Pieter van Dam's Beschryvinge van de Oostindische Compagnie*[1] and *Generale Missiven van Gouverneurs-Generaal en Raden aan Heren XVII der Verenigde Oostindische Compagnie*[2]. These corpora were supplemented by excerpts from *Memories van overgave van gouverneurs van Ambon in de zeventiende en achttiende eeuw*[11] to make the embeddings more robust. This combined corpus of 242.645 sentences spans the period between 1605-1788. Word embeddings were trained in gensim with the following parameters: {dimensions = 100, window = 5, min count = 3}[12].

3.3 CNN-BiLSTM-CRF

Our model uses a three-pronged approach (illustrated in Figure 1): The CNN branch extracts character-level features which are concatenated to word embeddings before moving through bidirectional LSTM and CRF layers.

The motivation for selecting this approach is its reproducibility on a variety of data and tasks, in this case, a sequence labelling task for historical Dutch. Traditional NERC systems make use of extensive feature engineering and pre-processing however, this was not necessary in our case, as the state-of-the-art CNN-LSTM-CRF architecture is able to learn these features by itself. [13].

We used a Bi-LSTM with CRF on top, using both past and future input features (Bi-LSTM) and focusing on sentence level (CRF)[14]. The LSTM layer is applied to take into consideration the word order sequence and learns long-range dependencies by incorporating a memory-cell[15]. It takes the name 'Bi-directional LSTM' as it learns the context of every sentence at every word bidirectionally: from left to right as well as from right to left[15].

The CRF learns the constraints of the IOB-label order. This is necessary because using the LSTM on its own would result in independent assumptions, which is not possible with IOB tagging. The CRF layer makes tagging decisions jointly and as a result we achieve a valid IOB sequence of tags [15].

The CNN extracts character representations of words and is concatenated with the word embeddings that will feed the LSTM [13]. This accounts for spelling inconsistencies and Out-of-Vocabulary (OOV) words.

To build this system, we adapted our code from the following Github page: 'UKPLab/emnlp2017-bilstm-cnn-crf'[16].

3.4 Evaluation of our custom model

The model was trained on a corpus of human annotated gold data, comprising 650 train sentences, 81, dev sentences and 81 test sentences which were tokenized and labeled in IOB-format. Despite a relatively small corpus of gold data to train on, the model yielded 0.69 precision, 0.74 recall, and 0.71 F1 scores.

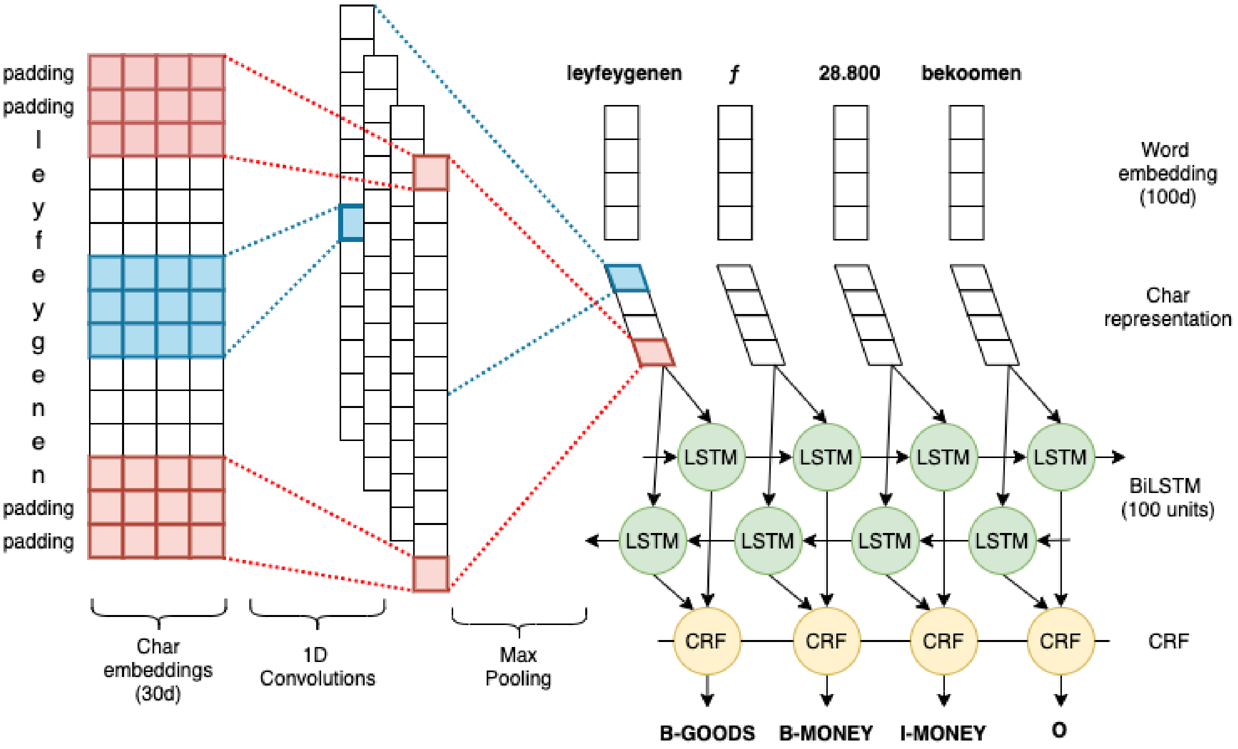

Figure 1: CNN-BiLSTM-CRF architecture.

	precision	recall	F1	support
ORG	1.00	0.87	0.93	15
LOC	0.83	0.93	0.87	67
PER	0.68	0.73	0.70	37
QUANTITY	0.72	0.67	0.69	27
DATE	0.54	0.71	0.61	21
SHIP	0.60	0.60	0.60	5
GOODS	0.53	0.60	0.56	30
MONEY	0.53	0.55	0.54	31
micro avg	0.69	0.74	0.71	233
macro avg	0.69	0.74	0.71	233

Table 3: Classification report.

Table 3 shows precision, recall and F1 scores broken down by named entity label. This classification report was created using the *seqeval* package[17]. The labels 'DATE', 'GOODS' and 'MONEY' score lowest. In Figure 2 the classification confusion matrix is shown. Per label we give an example of how this could be explained.

The dates are sometimes written in a convoluted way. E.g. 'den 28e der gemelte maand april' is at first sight not as clear as '28 april'. However, this was labeled correctly by the model. Goods are also harder to label, because it has fewer characteristics. In this way the model labeled 'zijde' as goods, but in this sentence it meant 'sight' instead of 'silk'. Given that the entities money, quantity and date contain mostly numbers, the label MONEY is sometimes attached to a number which isn't money, such as: 1200 (B-MONEY) a (O) 1500 (B-QUANTITY) inlanderen (O). The label LOC is sometimes given to an entity which should have been labeled as SHIP and vice versa. The confusion matrix in Figure 2 shows this. This is due to the fact that many VOC ships are named after a location, such as Amsterdam. The label ORG has a precision of 1. The recall is lower. Sometimes a location was labeled as ORG. The label LOC has a higher recall than precision. Some locations which should have been labeled as LOC were labeled as SHIP, mentioned earlier.

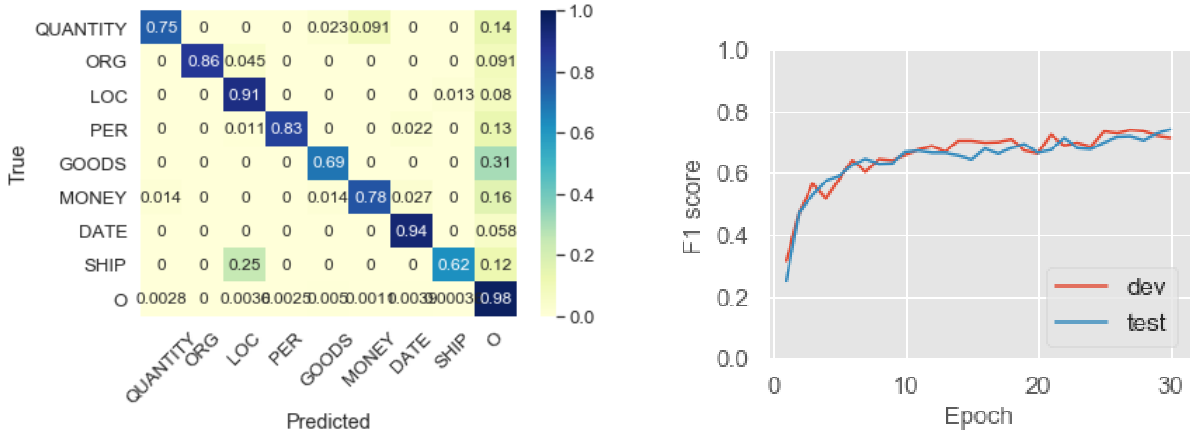

Figure 2: Confusion matrix (left) and Training history (right).

Figure 2 on the right shows the F1 scores of the development and test sets per epoch. The lines are close to each other, which means the model didn't suffer from over-fitting.

4 Timelines

4.1 Pipeline

To provide answers to the *What*, *Where*, and *Who* questions, we devised a pipeline to create three timelines. Figure 3 shows a schematic representation of our pipeline. In the following sections we will provide a more detailed explanation of how the timelines were constructed.

4.2 *What*: Goods

The goods component of the app extracts goods and quantities from texts, aggregates the quantities per decade, which is visualized on a timeline.

We passed the *Generale Missiven* texts through our model to extract QUANTITY and GOODS entities. Adjacent QUANTITY-GOODS pairs were linked to the decade corresponding to the date of the missive from which they were extracted, obtained through the *GM* metadata. Next, the quantities were split into numerical amounts and measurement units.

The resulting (decade, goods, amount, unit) tuples were subsequently passed through a series of filters, conversions, and clustering, before visualization.

First, because of the wide variety of spelling inconsistencies, we applied Damerau-Leveshtein fuzzy string matching, which accounts for deletions, insertions, substitutions, and transpositions. The resulting matrix of string similarities between goods was passed through Scikit-Learn's DBSCAN to cluster goods with inconsistent spellings.[18, 19]

Because there were no prevailing measurement standards in the VOC era, many different units were used, e.g. 'pikols', 'taël', 'kati', etc. We converted the amounts and units to Amsterdam pounds (lb.) based on the conversion rates described in *Huygens' VOC-glossarium*. [20] All QUANTITY-GOODS pairs for which there was no conversion rate to lb., were removed. While this filtering step removed a sizeable chunk of goods, it would have been impossible to aggregate goods by weight if the measurement units were not unified.

The app allows the user to specify the top n goods to be visualized by taking the sum of all quantities (in lb.) per decade. In Figure 4 we show the five most frequently traded goods.

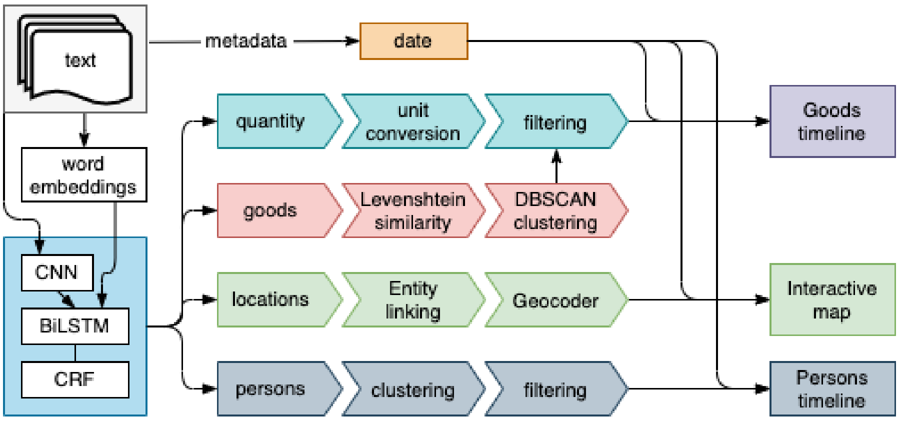

Figure 3: Timeline pipeline

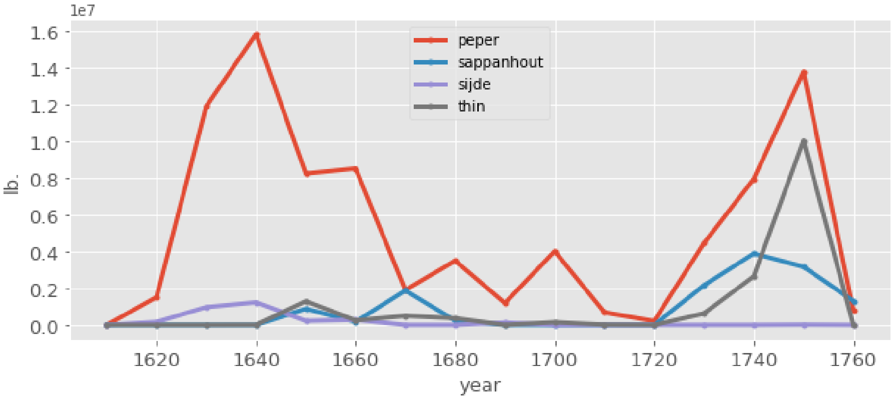

Figure 4: Most frequently traded goods over time.

4.3 *Where*: Locations

For the visualisation of the location component of the app, we chose to create a map with the LOCATION entities represented as way-points. Just like the goods component, we wanted to show those locations over time, which is why the map has a timeline.

One of the problems that we encountered is that the location names changed a lot since the moment these were written and there was no reliable way to convert them to the modern names automatically. Hence, we decided to create a dictionary of the 200 most occurring location names ourselves.

The way-points on the map are connected to the original text, which is publicly available. When the way-point is clicked on, it will show the original name of the location as a link, which will direct the user to the original text.

4.4 *Who*: Persons

The persons component of the app allows the user to extract the top n most frequently mentioned persons in the texts and visualize their rise to prominence and decline over the years based on normalized mention frequencies.

From the entities labeled by our system, all PER entities were extracted and linked to the date corresponding to the missive from which they were extracted. We decided not to correct for spelling inconsistencies by applying Damerau-Levenshtein fuzzy string similarity clustering, because it would take 72 hours to execute. The clustering option is however available in the script.

Finally, the number of mentions per person per decade was normalized to correct for the number of available missives per decade. Figure 5 shows the generated timeline for the top 5 most frequently mentioned persons.

5 Conclusion

In this project we aimed at creating an application that creates a timeline of historical data of the Verenigde Oost-Indische Compagnie. Therefore, we tested a NERC system trained on contemporary Dutch and a CNN-BiLSTM-CRF architecture trained on VOC data. Evaluation scores show that our own custom VOC-model outperforms the first model trained on modern Dutch. Besides the higher evaluation scores, another advantage of our own model was that we could choose our own NERC labels. We added labels for ships, goods and money. Hence, the timeline is created with the output of the second model. The timeline shows traded goods over time (aggregated per decade); most mentioned persons per missive over time ((aggregated per decade); locations on a map and connected to the missive where the location is mentioned.

6 Discussion

Historical data is different to data based on current Dutch. Due to this, there were some difficulties that would not occur when working with contemporary data. Sometimes, OCR resulted in non-existing words and this data could not be used. Other times, the sentence tokenizer would split the sentences incorrectly due to the way some of the original punctuation is used in an entity such as money. While the NERC model was able to handle spelling inconsistencies because of the CNN component and provide accurate named entity labels, the spelling inconsistencies still proved difficult to work with further down the pipeline. We attempted to solve this by applying Damerau-Levenshtein fuzzy string matching however, this method is computationally expensive for very large sets of data, and the resulting clusters are not always accurate. For example, the Levenshtein distance between 'gout' 'goud' and 'gout' 'hout' is exactly the same (one substitution), but phonetically the former pair would be closer in distance than the latter. Levenshtein distance treats all substitutions as equal.

Moreover, word embeddings don't capture polysemy, it would be interesting to test whether we could obtain better results by incorporating BERT embeddings (Bidirectional Encoder Representations from Transformers), which do allow for polysemy [21, 22].

Besides this, it was not possible to get enough training data due to time constraints. Therefore, we could not evaluate each other's annotations or calculate the inter-annotator agreement. This would have been useful to do, because we had difficulties to achieve inter-annotator agreement. Unfortunately, we could not use the silver data we created via fuzzy matching due to two main reasons. Firstly, we observed multiple incorrect named entity labels in the silver data and this was due to discrepancies present in the indices data. For example, the same keywords had different entity labels. Secondly, we observed very low quality of match results between keywords from indices data and missive tokens. This was because we used a fuzzy matching with one letter replacement. This approach results in incorrect matching of words with less than 4 letters thus creating more incorrect labelling in silver data.

As for generating the goods timeline, we excluded all the goods which were not convertible to lb., such as 'kisten' *chests* and 'sacken' *sacks*. We decided to filter out these goods, because it is impossible to sum quantities with differing units. Specialists who have more knowledge about the weights and measurements used in the VOC time period, could update the unit conversion dictionary in order to improve the timeline.

Lastly, It would have been ideal if the application could have generated the map and timeline automatically. So, for further research we

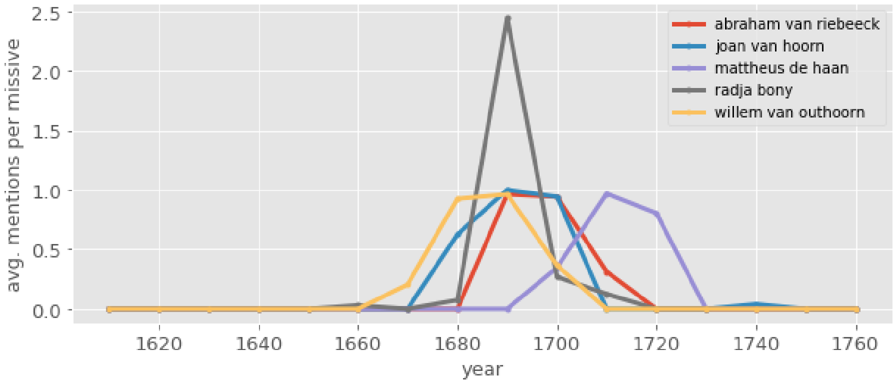

Figure 5: Most frequently mentioned persons per missive over time

would increase the ease of use by making an automatic connection between the NERC system, the map and the timeline.

References

[1] H. Stapel, "Pieter van dam's beschryvinge van de oostindische compagnie, 7 vols," *The Hague: Martinus Nijhoff*, vol. 1954, 1927.

[2] W. P. Coolhaas, *Generale missiven van gouverneurs generaal en raden aan Heren XVII der Verenigde Oostindische Compagnie*, vol. 258. M. Nijhoff, 2007.

[3] G. Rutten, A. Krogull, and B. Schoemaker, "Implementation and acceptance of national language policy: the case of dutch (1750–1850)," *Language Policy*, pp. 1–21, 2019.

[4] R. Cotterell and K. Duh, "Low-resource named entity recognition with cross-lingual, character-level neural conditional random fields," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 91–96, 2017.

[5] J. Xie, Z. Yang, G. Neubig, N. A. Smith, and J. Carbonell, "Neural cross-lingual named entity recognition with minimal resources," *arXiv preprint arXiv:1808.09861*, 2018.

[6] S. Schweter and J. Baiter, "Towards robust named entity recognition for historic german," *arXiv preprint arXiv:1906.07592*, 2019.

[7] D. Hupkes and R. Bod, "Pos-tagging of historical dutch," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 77–82, 2016.

[8] P. Vossen, "Lab4a nerc using a bidirectional, long-short-term-memory(bilstm) and conditional random fields (crf)," 01 2020.

[9] N. Garg, L. Schiebinger, D. Jurafsky, and J. Zou, "Word embeddings quantify 100 years of gender and ethnic stereotypes," *Proceedings of the National Academy of Sciences*, vol. 115, no. 16, pp. E3635–E3644, 2018.

[10] J. Zhao, T. Wang, M. Yatskar, R. Cotterell, V. Ordonez, and K.-W. Chang, "Gender bias in contextualized word embeddings," 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)*, (Minneapolis, Minnesota), pp. 629–634, Association for Computational Linguistics, June 2019.

[11] G. J. Knaap, *Memories van overgave van gouverneurs van Ambon in de zeventiende en achttiende eeuw*, vol. 62. Verkrijgbaar bij M. Nijhoff, 1987.

[12] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. http://is.muni.cz/publication/884893/en.

[13] M. Xuezhe and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf,"

[14] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[15] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition,"

[16] N. Reimers and I. Gurevych, "Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Copenhagen, Denmark), pp. 338–348, 09 2017.

[17] H. Nakayama, "seqeval software on github."

[18] G. Fairchild, "pyxdameraulevenshtein implements the damerau-levenshtein (dl) edit distance algorithm for python in cython for high performance.," 2013.

[19] M. Hahsler, M. Piekenbrock, and D. Doran, "dbscan: Fast density-based clustering with R," *Journal of Statistical Software*, vol. 91, no. 1, pp. 1–30, 2019.

[20] M. Kooijmans and J. Schooneveld-Oosterling, *VOC-glossarium: verklaringen van termen, verzameld uit de Rijks Geschiedkundige Publicatiën, die betrekking hebben op de Verenigde Oost-Indische Compagnie.* Instituut voor Nederlandse Geschiedenis, 2000.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[22] G. Wiedemann, S. Remus, A. Chawla, and C. Biemann, "Does bert make any sense? interpretable word sense disambiguation with contextualized embeddings," *arXiv preprint arXiv:1909.10430*, 2019.

A Annotation guidelines

Named Entity types: {DATE, LOC, PER, ORG, SHIP, GOODS, QUANTITY, MONEY}

Follow the IOB-format in labeling named entity spans, i.e. the first (or only) token of a named entity span is labeled 'B', and all subsequent tokens within the same span are labeled 'I', followed by the type of entity. Tokens which do not belong to a named entity span are labeled 'O'.

Specific guidelines per named entity type are given below.

1. DATE
   - Calendar dates in DD-MM-YYYY, DD-MM-YY, DD-MM, YYYY formats, e.g. '18-04-1723', '18-04-23', '18-04', '1723'
   - Calendar dates in text format including (if present) the article 'den', e.g. 'den 18e april 1723', 'den 18e april', '18 april', 'den 18e', 'april', '1723'.
   - Mentions linking to a date or a date range, e.g. 'anno passato', 'vandaag'.
   - Multiple dates connected by 'en' are labeled as a single entity span, e.g. 'den 18e en 19e april' will be labeled B-DATE, I-DATE, I-DATE, I-DATE.
2. LOC
   - Geographical locations, e.g. 'Nederlands-Indië', 'Gujarat', 'Java'.
   - Exclude adjectival forms of locations, e.g. 'Chinese'.
3. PER
   - Named persons, e.g. 'Johan van Oldenbarnevelt'
   - Include words such as 'van', 'van den' if they are part of the name.
   - Include titles, e.g. 'Koning Osman', 'Koning van Attingola', 'Gouverneur van Java'.
   - Exclude titles without any reference to the specific person, e.g. 'de koning'.
   - Exclude anaphora.
   - Exclude groups of people, e.g. 'de Javanen'
4. ORG
   - Organization names.
   - Include 'de Compagnie' as an abbreviation of 'Vereenigde Oostindische Compagnie'.
5. SHIP
   - Ship names
   - Exclude ships without reference to a specific name, e.g. 'het schip'
6. GOODS
   - Traded, acquired, or sold goods, e.g. 'peper', 'nagelen', 'sappanhout'.
   - Include serfs: 'leyfeygenen'.
   - Exclude quantities.
7. QUANTITY
   - Quantities of traded goods, e.g. '400', '17.547 lb.', '12.000 stux'.

  - Include the unit, e.g. ’17.547 lb.’, ’12.000 stux’.
  - Exclude quantities of money.

8. MONEY

  - Monetary amounts, e.g. ’*f*10.618,16,-’, ’8593 rd.’
  - Include amounts with 2 commas (which likely represents different coins), e.g. ’*f*76.522,11,8’
  - Include the ’,-’ indicating zero, e.g. ’*f*10.618,16,-’
  - Include different currencies and currency symbols, e.g. ’14.500 ropia’, ’*f*10.618,16,-’, ’8593 rd.’