Computing Meaning in Premodern Text Corpora

B.Eng.602 (formerly B.EP.301) / B.Eng.631 (formerly B.EP.11b) / M.EP.02b / M.EP.05b / M.EP.05d / M.DH.01 / M.DH.12

revision of January 17, 2023

Term:	Winter 2022/2023	**Instructor:**	Dr P. S. Langeslag
Time:	Tuesdays 10:15–11:45	**Office:**	SEP 2.306
Room:	KWZ 2.602	**Office hours:**	By appointment (office/online)
Credits:	See module description	**E-mail:**	planges@uni-goettingen.de
Prerequisites:	See module description	**Course website:**	langeslag.uni-goettingen.de/cmptr

This syllabus comprises an Overview (p. 1), a Schedule (p. 2), and an annotated Bibliography (p. 13).

Overview

Course Description

One of the most exciting avenues of natural language processing (NLP) is the ability to infer the meanings of words, sentences, and documents by processing the relations between them. This approach is particularly potent when applied to Big Data corpora the likes of born-digital English, though processing power becomes a significant limiting factor. With smaller premodern corpora, by contrast, the bottleneck tends to be statistical validity, but most tasks can be carried out on a consumer laptop.

In this course, we'll investigate what's possible when we turn the latest algorithms on the earliest English text corpora. You'll learn to deploy a selection of Python libraries aimed at statistical, semantic, and sentiment analysis, and we'll explore the added value of processing text and translation in parallel. We'll plug language data into a machine-learning classifier to try and distinguish between different kinds of content. We'll furthermore familiarize ourselves with the challenges of lemmatization, a high-dimensional classification task traditionally requiring a heavy burden of manual labelling, but we'll consider what existing data may be mined to lighten the load, as well as how machine learning may help solve the traditionally hard classification problem of homography. We'll close out the term identifying opportunities for pushing the study of Old English (or comparable fields) forward in a major way employing methods that are within your reach to wield in the context of a term paper, thesis, or longer-running research project.

The course assumes no prior programming knowledge, and students in modules outside the English Department in particular are not expected to have a prior knowledge of Old English.

Assessment

For students of B.Eng.602 (formerly B.EP.301), an **exam (60 minutes, 14 February 2023 at 10:00 sharp)** covers the seminar material. The associated lecture course will be assessed in a separate exam to be administered by its convenor; you'll want to register separately for each of the two courses and exams. Detailed specifications of the seminar exam will follow in the second half of the term. Weekly readings and exercises will serve alongside active class participation to prepare you for the exam.

Students of B.Eng.631 (formerly B.EP.11b), M.EP.02b, M.EP.05b, and M.EP.05d will write a **term paper** (**due 31 March**; see module description for length requirement, and here for specifications and topics) either on the topic of NLP itself or relying on its methods for the study of a premodern corpus. Students of M.EP.02b also write the above-mentioned lecture exam; the other graduate modules do not have an exam.

Students of M.DH.01 and M.DH.12 will each give a **presentation** on a topic to be agreed on with the instructor, and write up their findings in a **project report (due 31 March)**. Regular attendance is a prerequisite. For M.DH.01 the presentation is to be c. 20 minutes and the write-up c. 10 pp., for M.DH.12 both requirements are 50 percent longer. See here for specifications.

Required Texts and Resources

We will rely on excerpts from three textbooks:

1. Bird, Klein, and Loper, *Natural Language Processing with Python* (revised online edition);
2. Lane, Howard, and Hapke, *Natural Language Processing in Action* (excerpts on Stud.IP); and
3. Jurafsky and Martin, *Speech and Language Processing* (3rd ed. online draft);

as well as a selection of scholarly articles (Stud.IP) and online resources.

Diversity

This course is run with the understanding that students bring a variety of backgrounds into the classroom in such domains as socioeconomics, appearance, culture, religion, ability, gender, age, home/family situation, and sexual identity. With different backgrounds come different needs and sensitivities. If you feel your needs or those of a fellow student require special attention or are being compromised, please feel free to make this known to me by whatever channel seems most appropriate. (For more serious concerns, the Faculty and the University each have their own points of contact as well.) I will treat all requests seriously and with confidentiality, and will seek to make accommodations within my abilities and reason. At the same time, you too owe it to your fellow students to treat them with respect regardless of their background and identity. Do not stand in the way of anyone's well-being.

Tutorials

If you wish to learn, or improve your reading knowledge of, Old English, look for tutorials and reading groups on Stud.IP. If capacity allows, the tutorials for the Introduction to Medieval English Literature and Culture are open to you, as is the student-run Old English Reading Group. For a broader introduction to Python, please look for current course offerings on Stud.IP.

Schedule

Please prepare the following readings and do the following work *ahead* of the corresponding class, taking notes for in-class discussion.

Your answers to any study questions are *not* to be submitted in writing; instead, these questions help you prepare for in-class discussion, while also guiding your exam preparation if applicable. You won't be able to answer every last question, and some don't have "correct" answers at all but serve to stimulate reflection.

Week 1 (25 Oct): Concepts

Read:

- Searle, "Minds, Brains, and Programs" (8 pp.; Stud.IP)
  Takeaway: A foundational essay on the distinction between processing and understanding.
  Reading notes:
  1. Searle incorporates replies to his position into his article at pp. 419–424. These are an integral part of the text proper, which concludes on p. 424; the pages that follow are, first, actual peer responses (424–450); then, Searle's response to these peer responses (450–456). Please read the actual article, including the integrated responses I–VI and Searle's answers to them (i.e. 417–424); the remainder is optional.
  2. Searle references the Turing test (419 and passim). This test asks humans to evaluate whether they are interacting with another human or with a chatbot. If a chatbot is mistaken for a human agent by a certain proportion of test subjects, it has passed the Turing test, meaning it shows intelligence indistinguishable from that of a human (within the parameters of the test).

  Study question:
  1. If machine understanding is unattainable, in what ways do you suppose computing may contribute to semantic insights? Come up with as many different approaches as you can.

Further reading (optional):

- Chalmers, "The Puzzle of Conscious Experience" (7 pp.; Stud.IP)
  Takeaway: Asserts that consciousness cannot be reduced to mechanical functions but involves subjective experience.
  Reading note:
  1. This article has little to do with computing, but it gives a sense of the wild west that was the field of philosophy of mind in the decades after Searle's article. I include it here just in case the puzzle of consciousness has piqued your interest.

Week 2 (1 Nov): Python and NLP

Read:

- Langeslag, "JupyterLab" (4 pp.)
  Takeaway: Basic instructions on how to navigate the environment you'll rely on for your homework.
- Bird et al. ch. 1: "Language Processing and Python" (33 pp.)
  Takeaway: Walks you through querying the textbook's "Examples" corpus using the Python interpreter, and teaches the basics of Python and some widespread domains of NLP along the way.
  Reading notes:
  1. Some chapters (but not all) in the online edition of this textbook omit the chapter reference in headings; thus "§1.1.3" in these notes appears as section "1.3 Searching Text" in the HTML.
  2. The exercises at the end of each chapter are optional; we won't discuss them in class. I do, however, urge you to try out all the book's code examples ("listings") given over the course of each chapter's main content as you do your weekly readings.
  3. We are working in a text-only interpreter, so NLTK's downloader is not graphical as in the textbook. The easiest way to download the textbook materials is simply to enter `nltk.download('book')` instead of `nltk.download()`, then skip to importing.
  4. Please note that `nltk.FreqDist` really just reproduces `collections.Counter`, so the two may be used interchangeably. Just make sure to call it under the name by which you've imported it.

  Study questions:
  1. §1.1.4: Keeping in mind that corpora may be in any modern or premodern language, what language features does the `lexical_diversity()` function fail to account for?
  2. §1.5.1: For each of the examples of word disambiguation here given, can you formulate `if ... then`-disambiguation rules? Use shorthand prose or pseudo-code, not actual Python at this stage.

Do:

1. Look for Python tutorial channels on your choice of video streaming platform, subscribe to your favourite, and come prepared to recommend it in class.

Further listening (optional):

- Real Python Podcast episode 119: "Natural Language Processing and How ML Models Understand Text" (58m)
  Takeaway: A podcast episode aimed at Python learners that introduces the basics of NLP, including some of the classic machine learning approaches.

Week 3 (8 Nov): Handling Plaintext Corpora

Read:

- Bird et al. §§2.1.9, 3.1–3.2, 3.4–3.5 (26 pp., from ch. 2: "Accessing Text Corpora and Lexical Resources" and ch. 3: "Processing Raw Text")
  Takeaway: Explains how to import text data (ch. 2) and use stock functions and regular expressions to manipulate strings (ch. 3).
  Reading notes:

1. §3.1 "Electronic Books": Though Project Gutenberg is thankfully accessible from Germany again, the files' front and back matter has changed somewhat since the current revision of the book was made available. Thus to locate the start of the back matter you will want to run `raw.rfind("*** END")` rather than `raw.rfind("End of Project Gutenberg's Crime")`.
2. Under §3.1 "Reading Local Files," if you are following along with the examples, the easiest way to create a file in JupyterLab is to select the Text File icon in the launcher category "Other." You can use the "right-click" context menu to rename `untitled.txt` into something more memorable after saving.
3. Also under §3.1 "Reading Local Files," the `U` flag on Python's stock function `open()`, for universal newline mode, has been deprecated and superseded by an option `newline=None`, which is set by default, as is `'r'`; so just use `f = open('document.txt')`. Note that you do have to repeat the `open()` command after running the `.read()` method, as the garbage collector closes it at this point! As you get more Python under your belt, you'll learn about ways of retaining information that won't require you to reopen files.
4. §3.3 is a detailed treatment of text encoding solutions in Python. As long as we ensure we only work on UTF8 systems and with UTF8 files, we don't need to worry about this, but do refer back to this section if you run into issues with non-ASCII characters.

- Lane et al. §§2.2.4–2.2.5 (15 pp., from ch. 2: "Build Your Vocabulary (Word Tokenization)")
  Takeaway: Covers n-grams and normalization, including stop words, stemming, and lemmatization.
  Reading notes:
  1. Erratum: on p. 53 for "would they chose," read "would they choose."
  2. In this part of the textbook, the code listings are still very short and straightforward. You may choose to reproduce them just to improve your grasp of Python and these concepts.
  3. Note how, on p. 61, the authors assume that lemmatization will always "significantly reduce the precision and accuracy of your search results." This is because they are pricing the error inherent in the use of the generalized rules for stemming and lemmatization into the mechanism. If we have a reliable lemmatizer, of course, we are *increasing* the reliability of our data. We'll look at this in January.

  Reading questions:
  – The authors write that a typical stop word list contains c. 100 items (p. 52). Do you think this holds equally true for languages like Modern English and Modern German (and consequently Old English)? Why? What about the difference between Modern English and Cantonese?
  – Thinking about Modern English, or Modern German, or Old English if you can, what parts of speech (noun, pronoun, adjective, article, verb, adverb, preposition, conjunction) would be easy to stem, and which hard? Why?

Do:

1. Your working copy of the `git` repository includes a corpus of Old English preaching texts called ECHOE in the folder `echoe/`. Return to Bird et al. §2.1.9 ("Loading Your Own Corpus"), define `corpus_root` as `'echoe'`, load `.*` into `PlaintextCorpusReader` as described in the instructions there, and inspect the `fileids()` so you know what documents are available to call.
2. Now return to Bird et al. §1.1.4 ("Counting Vocabulary") and determine for a selection of ECHOE documents how their lexical diversity compares to some of the documents included with `nltk.book` (`text1`, `text2`, etc.), running the necessary operations on the `words` property of your object, e.g. `len(wordlists.words('394.11.txt'))`; or, better, define variables like `document = wordlists.words('394.11.txt')` and run your `lexical_diversity` function on them. (`wordlists` is an odd choice of variable name for what is really a text corpus; you may want to choose something more appropriate, such as `echoe`, instead.) What strikes you about the lexical diversity of these Old English preaching texts compared to those shipped with the textbook? How do you explain your findings?
3. Now head to Bird et al. §1.3.1 ("Frequency Distributions"), add `from nltk import FreqDist` to your imports, and plot a non-cumulative graph for the 50 most frequent word forms in three different texts. Save your notebook for in-class discussion. What is the most frequent type? What could be done to clean up the results? Now compare your graph with the cumulative plot from the textbook. Why do you suppose the textbook said to generate a *cumulative* graph?
   Tip: From this point on, we'll work with ECHOE almost every week. If you want to try our methods on a different corpus, nothing stops you from loading your own corpus into JupyterLab, or into a local installation of Python. Just make sure to format and normalize your corpus thoroughly (cf. Lane et al. §2.2.5), which for ECHOE I have done ahead of time to make your homework less cumbersome.

Week 4 (15 Nov): TF-IDF

Read:

- Lane et al. ch. 3: "Math With Words (TF-IDF Vectors)" (27 pp.)
  Takeaway: An introduction to the mathematical magic behind the computational analysis of relevance.
  Reading notes:

  1. From this point onward, there is no need to reproduce the code illustrations in this textbook. Instead, focus on understanding the concepts and how the code works. If you do want to try out some of the code, the authors provide it in Jupyter Notebook format here; or open a terminal window and run `git clone https://github.com/totalgood/nlpia.git` in a terminal window from your home directory to download all the textbook's examples along with the source code of the `nlpia` package. Make sure to install the necessary packages (`pip install nlpia scipy sklearn`) before running the code examples.
  2. Just to make sure you get this right: Zipf's Law states that a word's frequency approximates one over its frequency rank ($\frac{1}{rank}$), multiplied by the number of occurrences of the most frequent word. So given that *the* ranks first and *of* second in a typical Modern English corpus, a document counting 100 instances of the former should have about $100 \cdot \frac{1}{2} = 50$ of the latter.
  3. Errata:
     (a) On p. 74, the authors assert that "normalized term frequency. . . [is] the word count tempered by how long the document is." This is the accepted definition of normalized TF. By their own method, however, the TF is in fact placed against the length of the list of *types*, not the list of tokens or document length! To see the difference, compare the output of `len(bag_of_words)` with that of `len(tokens)`. In their exercise on p. 87 the authors correctly measure against the list of tokens. Then on p. 90 the mathematical expression $count(d)$ is imprecise, because you are not counting the document, but rather the number of tokens it contains.
     (b) On p. 77 for "This collections" read "This collection."
     (c) On p. 85, for "2" in "You'll learn about part-of-speech tagging in chapter 2," read "11."
     (d) On p. 87, ignore the object names `intro_doc` and `history_doc` given in the text body, as they don't recur.
     (e) On p. 90, to access `log()` you'll have to import it from the `math` or `numpy` library.
     (f) On p. 91 `query_vec` is defined twice where once would have sufficed.

  Study question:

  1. Reflecting on TF-IDF in conjunction with Searle's Chinese room argument, what can statistics do towards inferring meaning?

Do:

1. Now that you have seen how to conduct TF-IDF analyses both from scratch (Lane et al. §§3.4.1–3.4.3) and through prepared packages (§3.4.3), take on a set of three ECHOE documents of your choice (e.g. three versions of the Assumption of Mary: `032.11`, `048.54`, and `382.13`; or of *Sermo Lupi*: `068.04`, `049B.40`, and `331.27`; or of Bethurum 17: `144.05`, `331.26`, and `331.30`; or three randomly chosen documents. See instructions for week 3 on how to access ECHOE; then to line up your texts, define a list as under Lane §3.2 but along the lines of `docs = [echoe.raw('032.11.txt'), echoe.raw('048.54.txt'), echoe.raw('382.13.txt')]`. Then create a vector matrix following the instructions in Lane §3.4.3. Finally, determine which two of your three documents are most alike as follows:
```
>>> from sklearn.metrics.pairwise import cosine_similarity
>>> cosine_similarity(model)
```
If your chosen documents are versions of the same text, inspect their respective length and determine whether differences in length correlate with cosine similarity. You may also be able to inspect the files side by side at echoe.uni-goettingen.de/testing/ to get an eyeball handle on similarity.

Week 5 (22 Nov): Classification

Read:

- Bird et al. §§6.1.1–6.1.5, 6.2.1, 6.3 (16 pp., from ch. 6: "Learning to Classify Text")
  Takeaway: A hands-on lesson in classification tasks using machine learning.
  Reading notes:
  1. Remember to `import nltk` before reproducing the chapter's code.
  2. Take the "Your Turn" exercise of §6.1.1 seriously: this is where you acquire the skills to adapt a classifier to your own needs!
  3. Tip: if you `import string` before starting the `for` loop in example 6.1.2, the line `for letter in 'abcdefghijklmnopqrstuvwxyz':` may be more comfortably expressed `for letter in string.ascii_lowercase:`.

Do:

1. Return to the gendered names exercise of Bird et al. §§6.1.1–6.1.2 and repeat it for names appearing in documents in England up to the time the Domesday survey was completed (1086). To this end, load `pase/female.txt` and `pase/male.txt` instead of the files supplied with the textbook corpus. (Load and tokenize them using e.g. `male = open('pase/male.txt').read().splitlines()`; you can then use syntax like `for name in male`.) What features prove to be the most informative for these (modernized!) names? How accurate is your classifier of Old English names relative to the textbook's classifier of present-day names? What do you know about the way Old English names were chosen, how do you suppose these lists were obtained and normalized, and how do these facts complicate our ability to analyze them for gender in quite the same way as the modern lists? Can you think of other machine-readable features that might be more revealing?
2. The files `pase/female-dupes.txt` and `pase/male-dupes.txt` are like the files mentioned under (1) above, but without the elimination of duplicates; in other words, these add up to a full index (give or take) of named individuals whose names have come down to us from or via pre-Conquest England, but stripped down to just their given names, one per line. What can you learn from these data about relative frequency? (Hint: you will want to research the `Counter` module, which you can import `from collections`.) And can you confirm whether Zipf's law applies to the frequency of recorded Old English names?

Further reading (optional):

- Bird et al. §§6.4–6.7 (14 pp.)
  Takeaway: The remainder of the chapter offers further detail on methods used in classification. Well worth a read if you intend to undertake any such work yourself!

Week 6 (29 Nov): Topic Modelling

Read:

- Lane et al. §§4.1–4.3.5 (26 pp., from ch. 4: "Finding Meaning in Word Counts (Semantic Analysis)")
  Takeaway: A challenging introduction to how meaning may be computed.
  Reading notes:
  1. Here too, just try to understand the concepts, and the code as well as you can; don't worry about reproducing the code (but it's here). Given the complexity of the material, you'll want to read this excerpt more than once.
  2. Errata: footnote 13 refers to a chapter of Jurafsky and Martin's that is currently not available; in Figure 4.3 for "TE-IDF" read "TF-IDF."

  Study questions:
  1. §§4.1.1–4.1.2: Having read this section as well as Lane et al. ch. 3, can you draw up a list of advantages and disadvantages of relying on TF-IDF vectors as a semantic model?

2. §§4.1.4, 4.2: Can you sum up how latent semantic analysis works in under twenty words?

Do:

1. Study, then run `lsa.ipynb` from the repository.
2. What strikes you about the terms that the respective models choose as the building blocks for their topics?

Further reading (optional):

- Lane et al. §§4.4–4.8.2 (30 pp., from ch. 4: "Finding Meaning in Word Counts (Semantic Analysis)")
  Takeaway: The remainder of the chapter offers further detail on latent semantic analysis, with emphases on principal component analysis and LDiA.
  Reading note:
  1. This excerpt is not supplied on Stud.IP in view of copyright restrictions.

Week 7 (6 Dec): Word Embeddings

Read:

- Jurafsky and Martin ch. 6: "Vector Semantics and Embeddings" (31 pp.)
  Takeaway: An accessible, if occasionally maths-heavy, introduction to inferring meaning from the relationships between the words in a corpus.
  Reading notes:
  1. If you are struggling with the chapter, try reading Lynn, "An Introduction to Word Embeddings for Text Analysis" (3000 words) first.
  2. §6.5 revisits TF-IDF, but it exclusively treats it as a method for weighting word vectors. Your understanding of the technique may benefit from the reading of two complementary angles!
  3. Erratum: in formula 6.20, $p*_j$ should read p_j.
- Lynn, "Word Embeddings in Python With Spacy and Gensim" (2,000 words)
  Takeaway: A hands-on demonstration of word vector modelling.
  Reading note:
  1. The functions here based at `model.similarity()` and `model.most_similar()` have been moved to `model.wv.similarity()` and `model.wv.most_similar()`.

Do:

1. Let's train a word vector model on ECHOE. Because these models require tokenized sentences as input, we'll create a list of sentences (delimited by linebreaks in ECHOE) prior to tokenization.
```
>>> import glob
>>> from gensim.models import Word2Vec
>>> corpus = []
>>> for i in glob.glob('/home/jovyan/cmptr/echoe/*txt'):
>>>       document = open(i).read().splitlines()
>>>       for sentence in document:
>>>             corpus.append(sentence)
>>> sentences = [[token for token in document.split()] for document in corpus]
```
2. We are now ready to configure and run our model:
```
>>> model = Word2Vec(sentences=sentences, min_count=1, vector_size=300, workers=2, window=5,
epochs=30)
```
   This will take twenty seconds or so.
3. Once training is complete (i.e. when the cell receives a sequence number), you can test the model using queries like the following:
```
>>> model.wv.most_similar('deofol')
>>> model.wv.most_similar('niht')
```

```
>>> model.wv.most_similar('mæden')
>>> model.wv.most_similar('wop')
>>> model.wv.most_similar('blis')
>>> model.wv.distance('mann', 'cild')
>>> model.wv.distance('wif', 'cild')
>>> model.wv.distance('mæden', 'cniht')
>>> model.wv.distance('wop', 'blis')
```

4. What strikes you about the results you are getting? Where do you see room for improvement? Are you able to improve your model e.g. by tweaking the values of `min_count`, `vector_size`, and `window`, or by changing the subsampling routine from the default `sample=1e-3` to e.g. `0` or `1e-5`, or by extending the list of common terms? What other queries can you think of that may be of particular value in gauging model accuracy?

Further reading (optional):

- Lynn, "An Introduction to Word Embeddings for Text Analysis" (3000 words)
  Takeaway: An accessible introduction to word vector modelling.
- Lane et al. ch. 6: "Reasoning With Word Vectors (Word2vec)" (36 pp.)
  Takeaway: A fuller introduction to Word2vec, GloVe, and the algebra underlying them.
  Reading note:
  1. This excerpt is not supplied on Stud.IP in view of copyright restrictions.

Week 8 (13 Dec): Sentiment Analysis

Read:

- Lane et al. §2.3: "Sentiment" (7 pp., from ch. 2: "Build Your Vocabulary (Word Tokenization)")
  Takeaway: Briefly demonstrates rule-based and machine-learning approaches to inferring a document's judgement value.
  Reading note:
  1. As with previous readings from Lane et al., there is no need to reproduce the code illustrations. Instead, focus on understanding the concepts and how the code works. But if you are interested in trying out the code from this section, the authors have made it all available for download as a Python source file here.

  Study questions:
  1. Can you think of any medieval text types that might be used as training corpora for sentiment analysis using a machine-learning approach?
  2. What are the main challenges in finding applications for sentiment analysis in premodern corpora?
- Jurafsky and Martin §§25.0–25.4.3 (10 pp., from ch. 25: "Lexicons for Sentiment, Affect, and Connotation")
  Takeaway: Discusses the use and generation of lexicons in the service of sentiment analysis.
  Reading note:
  1. The chapter prior to §25.3 is rather general; you can look it over quickly, and read more closely if and when you end up doing a project on sentiment. Please read §§25.3–4 more carefully.

  Study question:
  1. How would you go about obtaining or creating a sentiment lexicon for a language like Old English?

Do:

1. Read and study `sentiment.ipynb`, provided in the course repository.

Further reading (optional):

- The rest of Jurafsky and Martin ch. 25 (9 more pp.)
- Bessa, "Lexicon-Based Sentiment Analysis: A Tutorial" (2200 words)
  Takeaway: An accessible explanation of basic rule-based sentiment analysis.
  Reading note:

1. The practical implementation described in this blog post uses KNIME, a visual programming platform. This means you'll have to ignore mention of such environment-specific concepts as "nodes," unless, of course, you'd like to learn to use the environment.

- Sprugnoli et al., "*Odi et amo*: Creating, Evaluating and Extending Sentiment Lexicons for Latin" (9 pp.)
  Takeaway: Describes efforts to create Latin sentiment lexicons.
  Reading notes:
  1. A note on terminology: "gold standard" refers to a data set created with the help of human evaluation (cf. Bird et al. §5.4.4); "silver standard" data harmonizes data from various sources, including processes carried out without human evaluation, in this case by filtering the gold standard data through databases on synonyms, morphology, and orthographical variation. "Seed terms" are one or more annotated words taken as a starting point, from which the unannotated terms in the corpus are evaluated using machine learning by studying how these word forms relate to the seed words within the corpus. This process is called bootstrapping; see Jurafsky and Martin §21.2.3.
  2. To see the results of the authors' work, browse their sentiment lexicons on GitHub.

Week 9 (20 Dec): Off-the-Shelf Lemmatization and Integrated Pipelines

---

Read:

- Kyle P. Johnson, "CLTK Demonstration" (1683 sloc)
  Takeaway: A brief technical demonstration of how CLTK may be used off the shelf.
  Reading notes:
  1. As this demonstration was written as a Jupyter Notebook, you can execute the code as you read along, provided you are accessing the file within JupyterLab or on your own Jupyter server. (You can download all CLTK code, including notebooks, by running `git clone https://github.com/cltk/cltk.git` from your home directory in a terminal.) However, we may as well read along directly on GitHub: since the author has embedded his outputs in the file, we don't need to run the code ourselves to follow along. If you do want to try out the code, you'll have to reproduce the last line of code from the third cell in a terminal (in an appropriate working directory) to obtain the text data; the `%time` function in cell 12 only works in Ipython, but you can leave it out.
  2. You may skip the section on Greek; the Latin section suffices for demonstration purposes.
- Langeslag, "CLTK" (6 pp.)
  Takeaway: An attempt to make CLTK documentation both more accessible and more complete.
- From the spaCy documentation: "Language Processing Pipelines," up to and including "Analyzing Pipeline Components."
  Takeaway: An overview of spaCy's default pipeline.
  Reading note:
  1. spaCy is a more sophisticated NLP library, but it lacks support for historical languages. It may be that it has language-independent tools that suit your needs, but it doesn't have language models for Latin or Old English. In the list of supported languages, pay close attention to the "pipelines" column.

Do:

1. Now that you are in a position to lemmatize documents, investigate what happens to a document's lexical diversity (see weeks 2–3) if we run it through the lemmatizer first. Quantify the difference, then reflect on the implications of your findings. What research tasks justify lemmatization, and which are better off without? What makes lemmatization so slow? How would you proceed if you had to lemmatize more text than your system's working memory can handle?
2. Study `cltk.ipynb` in the course repository. Lemmatization works well for the document provided because I have manually normalized its language somewhat. Medieval Latin has the following peculiarities compared to standardized Classical Latin:
   - Classical <ae> often appears in the manuscripts as <e> (e.g. <eternus> for <aeternus>);
   - Consonantal <i> may appear in later manuscripts and some editions as <j> (e.g. <jussit> for <iussit>);
   - Classical <ti> is sometimes written <ci> (e.g. <laudacio> for <laudatio>);

- Intervocalic <h> is sometimes written <ch> (e.g. <michi> for <mihi>).

In addition, lemmatizers may expect documents to be normalized in two important respects:

- What is <v> in textbooks and normalized editions was in fact written <u> (e.g. <uita>) in the manuscripts, and in about half of modern text editions.
- Normalized <ae> may in the manuscripts, and some editions, be <æ> or <ę> (<æternus>, <ęternus>).

How would you address these issues in a lemmatization pipeline?

Week 10 (10 Jan): Ensemble Lemmatization

Watch:

- Langeslag, "Towards a Lemmatized Corpus of the Old English Homilies" (19m)
  Takeaway: A primitive version of my take on lemmatizing Old English prose.

Do:

1. Imagine you have access to all the preexisting data mentioned in the video under "Data Sets," as well as CLTK; i.e.:
   (a) The *Dictionary of Old English* headword list, associating each headword form with an identifier:
      - e.g. `headwords` is a dictionary, with entries like `652: 'and, ond'`;
   (b) Attested spelling data for forms associated with headwords in the A–K range: attested form, headword identifier, headword form, and number of times the attested form occurs in DOEC:
      - e.g. the dictionary key `attested['gode']` has as its value a list of tuples `[(38800, 'god', 33000), (12420, 'gōd noun', 1400)]`, in descending order of number of attestations;
   (c) DOEC lemmatization data for headwords in the range M–S: attested form, headword identifier, headword form, and number of times this form has been assigned to this headword:
      - e.g. the dictionary key `lemmatized['niwra']` has as its value a list with just one tuple `[(19805, 'nīwe adj.', 9)]`;
   (d) CLTK as a fallback option for the range A–Y (Old English lacks Z):
      - e.g. accessible as a function `cltk_lemmatize('form')`, returning a list of headwords in descending order of likelihood, but no identifier; thus e.g. `['god', 'gōd noun', 'gōd adj.']`.
2. Write a rough script (in pseudocode or prose) for how you would assign the forms in a new document to headwords. Come prepared to discuss your solution in class.
3. What role do you see for machine learning? Where could a trained classifier outperform a rule-based approach?

Week 11 (17 Jan): Word Sense Disambiguation

Read:

- Jurafsky and Martin ch. 23: "Word Senses and WordNet" (18 pp.)
  Takeaway: Introduces various complications in the relations between signifier and signified, then introduces the WordNet database and various approaches to word sense disambiguation.
  Reading note:

  1. In some respects rather basic for anyone with a linguistic background, this chapter is at least more accessible than the paper set for this week. Though it relies on knowledge bases that do not cover Old English, it also tackles disambiguation strategies more directly. Accordingly, §§23.4–23.7 are the sections that really matter; students with a background in linguistics can pass over §§23.1–23.2 with just a brief glance.

- Wunderlich et al., " 'God wat þæt ic eom god': An Exploratory Investigation into Word Sense Disambiguation in Old English" (10 pp.)
  Takeaway: Tests a selection of supervised approaches to sense disambiguation for Old English.
  Reading notes:

1. Erratum On p. 41, for "from from" read "from."
2. The paper mentions two contrastive approaches to classification: naive Bayes and maximum entropy. The former considers individual features without correlating them, which is a fast approach but often sufficiently accurate; the latter requires more processing but allows for the correlation of features (e.g. "if word X appears in the vicinity of the word under consideration, the odds of classification A being correct rise to 80%") and calculates the most probable classification given all the available correlations. These matters are explained in Bird et al. §§4.4–4.6.

Study questions:

1. Can you think of any approaches to Old English word sense disambiguation (from Jurafsky and Martin, or of your own devising) not or insufficiently considered in this paper? How would you go about this task?
2. Explain why Wunderlich et al. consider the Lesk algorithm unsuitable. Could any part of it be salvaged to be used on Old English nonetheless?
3. Reflect on the appropriateness of choosing Old English *bōc* "book" as the test subject.

Do:

As mentioned in last week's video, the Old English lexical form "god," as well as the inflected form "gode," is associated with three distinct headwords: the noun *god* "god," the noun *gōd* "good," and the adjective *gōd* "good." Given the classification toolkit we were handed in Bird et al. §6.1.1 ("Gender Identification"), the only remaining things we need in order to train a classifier are (1) a set of labelled data and (2) a well-chosen feature set, ideally consulting the token's context as in Bird et al. §6.1.5. I have provided the data by manually sorting some 70 percent of all ECHOE sentences containing the form "god" or "gode" into six files in the course repository under `disambiguation/`. I have also adapted the classifier demonstration from Bird et al. to accommodate the relevant data for two different classifiers (one for "god" and one for "gode") and set them up with a basic set of classification features in a notebook `disambiguation.ipynb`.

1. Run the notebook and study its code, making sure you understand exactly how it works.
2. Inspect the accuracy of the two classifiers, and see if you can feed them individual "new" sentences (e.g. from the saints' lives in the repo folder `aels/`, or manually copied in and normalized from Ælfric's *Catholic Homilies*).
3. Inspect the most informative features of both classifiers. How come some of the most informative features are empty values, such as `preceding3 = 'None'`?
4. Can you think of new features that might help improve the accuracy of one or both of these classifiers? How would you implement them? Keep in mind that all features have to be in Python dictionary form, with a unique key and a string value.

Week 12 (24 Jan): Bitexts

Read: Pick one (or read both) of the following texts:

- Tiedemann, *Bitext Alignment* pp. 59–81 (from ch. 5 "Word Alignment") (22 pp.; Stud.IP)
  Takeaway: An introduction to statistical models for identifying which tokens across a translated sentence represent each other. This is the more accessible of the two, and the better choice if you are not already steeped in neural network theory.
- Jurafsky and Martin §§13.0–13.7 (20 pp., from ch. 13: "Machine Translation")
  Takeaway: An introduction to modern models for word machine translation. This chapter relies on more prior knowledge about such mechanisms as attention and Transformers, but it better represents the current state of the field than the excerpt from Tiedemann. If you aren't familiar with the architecture but would prefer to learn about the latest models rather than a statistical approach, you may choose to read Jurafsky and Martin ch. 9, then ch. 13. Alternatively, watch a video like this one as a quickstart, then proceed to Jurafsky and Martin ch. 13.

Do:

1. Study `word_alignment.ipynb` in your working copy of the repository, then run it. Duplicate the file and replace the sentences with others from `word_alignment/vercelli1_latin.txt` and `vercelli1_oe.txt`; these have already been sentence-aligned, so lines of the same line number correspond between the two files. Do you have a sense of how well it does? Can we infer from the results how the aligner works under the hood?

Further reading (optional):

- Chen et al., "`Mask-Align`" (8 pp.)
  Takeaway: Introduces a new Transformer-based word aligner.
  Reading note:

  1. Relies on an understanding of how Transformers are used in Machine Translation models, e.g. as learned from Jurafsky and Martin chs. 9, 13.

- Christodouloupoulos and Steedman, "A Massively Parallel Corpus: The Bible in 100 Languages." (14 pp.; Stud.IP)
  Takeaway: Reports on the creation of the corpus in question.

Week 13 (31 Jan): Crosslingual Semantics

Read:

- At least §§4–6 (18 pp.) of Ruder et al., "A Survey of Cross-Lingual Word Embedding Models" (49 pp.)
  Takeaway: After summing up monolingual approaches, this article provides an overview of methods for aligning the word embeddings of corpora in two or more languages. You may skip at least §9, and any part of §§1–3 that feels redundant.
- Smith et al., "Aligning the fastText Vectors of 78 Languages" and the accompanying Jupyter notebook
  Takeaway: A practical demonstration of how two or more monolingual sets of embeddings may be aligned.

Further reading (optional):

- The conference paper accompanying Smith et al.'s code (8 pp.)
- Adams et al., "Cross-Lingual Word Embeddings for Low-Resource Language Modeling" (9 pp.)
  Takeaway: Scales down learned, bilingual lexicon-based continuous BoW approaches to low-resource conditions.

Do (optionally):

1. Using the above-mentioned notebook as your model, can you create Old and Modern English vector spaces and align them?

Week 14 (7 Feb): Opportunities in Old English NLP

Read:

- Torabi, "If (Not 'Quantize, Click, and Conclude'): {Digital Methods in Medieval Studies}" (18 pp.)
  Takeaway: Traces a medievalist's efforts to give user-friendly online tools a place in her scholarly workflow, and argues that they are valuable but only if used in conjunction with traditional humanist methods.
  Reading note:

  1. Where the author references Lexomics, this refers to Lexos, a tool well worth trying out; Lexomics is the project that developed it.

  Study questions:

  1. Using your knowledge of statistical methods and Old English (if any), can you comment on the author's failure to confirm a textual connection between Blickling Homily XVII and *Beowulf*?
  2. Using your understanding of statistical methods, can you comment on the author's conclusion that clustering methods cannot produce information on gender roles in literature?
  3. The author repeatedly refers to technological limitations. What's your take on the obstacles she encounters?

Do:

1. Evaluate at least half a dozen of the following online resources:

   (a) *Anglo-Saxon Penitentials*
   (b) *Cædmon's Hymn*: A Multimedia Study, Edition and Archive
   (c) CLASP: A Consolidated Library of Anglo-Saxon Poetry

   (d) *The Dictionary of Old English* and *Dictionary of Old English* Corpus
   (e) Early English Laws
   (f) *The Electronic "Beowulf"*
   (g) *A Thesaurus of Old English* and Evoke
   (h) The Digital Ælfric
   (i) *The Prosopography of Anglo-Saxon England*
   (j) The York–Toronto–Helsinki Parsed Corpus of Old English Prose (YCOE) (actual corpus downloadable here, through the Oxford Text Archive)

2. Judging by these projects (and the Torabi article), what problems seem to be baked into the scholarly culture of digital Old English studies thus far, particularly from an NLP perspective, and what would it take to overcome them?
3. Where do you see unexplored potential in the existing landscape of digital resources?
4. What underused approaches appeal to you personally? Where would you start?
5. How would you sell your ideas to a (humanities) scholarly funding body, such as DFG, ERC, AHRC, or SSHRC? How are the scholarly community, the state of our knowledge, and society at large helped by your proposed project?

Week 15 (14 Feb): `EXAM = [student for student in students if student in exam_takers]`

Bibliography

§1: Theory and Programming

Behnel, Stefan. "The lxml.etree Tutorial." https://lxml.de/tutorial.html.

Teaches the basics of navigating XML using Python.

Bird, Steven, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. Sebastopol, CA: O'Reilly, 2009. https://www.nltk.org/book/.

The official book of the NLTK Python library. The HTML version is freely available CC-BY-NC-ND, and unlike the print and Kindle editions it has been updated to work with Python 3 and NLTK 3, so the code is more likely to work. However, even the online edition relies on some deprecated functions.

Brunton, Steve. "Singular Value Decomposition," 2020–22. https://www.youtube.com/playlist?list=PLMrJAkhIeNNSVjnsviglFoY2nXildDCcv.

A playlist of 43 short explainers on SVD.

Deisenroth, Marc Peter, A. Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. Cambridge: Cambridge University Press, 2020.

Eisenstein, Jacob. *Introduction to Natural Language Processing*. Cambridge, MA: MIT Press, 2019.

An excellent but challenging because purely mathematical-theoretical survey of NLP methodology.

Johnson, Kyle P. *CLTK Notebooks*. https://github.com/cltk/cltk/tree/master/notebooks.

Demonstrations for the current 1.0 release, not as extensive as the tutorials for the deprecated 0.1.x release.

———. *CLTK Tutorials*. https://github.com/cltk/tutorials.

The most extensive CLTK documentation available, though written for a deprecated 0.1.x release.

———. *The Classical Language Toolkit* (CLTK)*: Legacy Docs*, 2019. https://legacy.cltk.org/en/latest/.

The somewhat sparse documentation for the deprecated 0.1.x release of CLTK.

Jurafsky, Dan, and James H. Martin. *Speech and Language Processing*. 3rd ed. draft. December 29, 2021. Accessed October 13, 2022. http://web.stanford.edu/~jurafsky/slp3/.

Online drafts of an outstanding textbook that does not teach programming but only the concepts and models used for NLP.

Lane, Hobson, Cole Howard, and Hannes Hapke. *Natural Language Processing in Action: Understanding, Analyzing, and Generating Text with Python*. With a foreword by Arwen Griffioen. Shelter Island, NY: Manning, 2019.

Dense and detailed, this textbook assumes a good deal of prior knowledge and isn't the best at explaining concepts. However, if you're willing to challenge yourself, it does cover a lot of ground and it drives home the knowledge using practical Python exercises, as well as explaining key concepts not covered in other textbooks.

Lynn, Shane. "An Introduction to Word Embeddings for Text Analysis." https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/.

A short and accessible introduction.

———. "Word Embeddings in Python With Spacy and Gensim." https://www.shanelynn.ie/word-embeddings-in-python-with-spacy-and-gensim/.

A hands-on demonstration.

Matthes, Eric. *Python Crash Course*. 2nd ed. San Francisco, CA: No Starch, 2019.

An affordable and well-received entry-level Python textbook.

Mitchell, Melanie. *Artificial Intelligence: A Guide for Thinking Humans*. New York: Farrar, Straus and Giroux, 2019. Reprinted by Pelican 2019.

An impressively well-written introduction to the methods of and concerns over AI.

Nielsen, Michael. *Neural Networks and Deep Learning*. http://neuralnetworksanddeeplearning.com.

An accessible open access online (HTML+Javascript) book introducing these concepts.

Perkins, Jacob. *Python 3 Text Processing with NLTK 3 Cookbook*. 2nd ed. Birmingham: Packt, 2014.

Walks readers through specific tasks, such as POS tagging or custom corpus importing, using NLTK.

"Real Python." https://realpython.com.

An excellent collection of tutorials and video courses. You can read (most?) text-based tutorials free of charge, though you have to make a free account after a while, and at least the video tutorials rely on a pricey membership model.

Real Python Podcast. 2020–. https://realpython.com/podcasts/rpp/.

An accessible podcast exploring many different aspects of Python.

Rudkowsky, Elena, Martin Haselmayer, Matthias Wastian, Marcelo Jenny, Štefan Emrich, and Michael Sedlmair. "More than Bags of Wods: Sentiment Analysis with Word Embeddings." *Communication Methods and Measures* 12 (2–3 2018): 140–157.

A good example of sentiment analysis in practice.

Sanderson, Grant. "3Blue1Brown." https://www.youtube.com/c/3blue1brown.

A calculus YouTube channel with excellent explainers on the maths behind programming, including machine learning.

Vasiliev, Yuli. *Natural Language Processing Using Python and spaCy: A Practical Introduction*. San Francisco: No Starch Press, 2020.

Concise but practical and valuable; focused on spaCy rather than NLTK.

"W3 Schools: Python Tutorial." https://www.w3schools.com/python/.

Accessible online tutorial. Not nearly as comprehensive as Real Python.

§2: Software and Resources

Bird, Steven, and Liling Tan. *NLTK: Natural Language Toolkit*. https://www.nltk.org.

A Python NLP library that remains in widespread use, though it is not as advanced as spaCy.

Johnson, Kyle P., Patrick J. Burns, John Stewart, Todd Cook, Clément Besnier, and William J. B. Mattingly. "The Classical Language Toolkit: An NLP Framework for Pre-Modern Languages." In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, 20–29. Association for Computational Linguistics, August 2021. https://doi.org/10.18653/v1/2021.acl-demo.3. https://aclanthology.org/2021.acl-demo.3.

The writeup of the conference presentation announcing the current version of CLTK, a Python library for the processing of dead languages.

Kleinman, S., M. D. LeBlanc, M. Drout, W. Feng, and C. Zhang. *Lexos*. V. 4.0, 2019. http://lexos.wheatoncollege.edu.

A set of online tools for NLP analysis.

Korolev, Slava. "Neural Network to Play a Snake Game." https://towardsdatascience.com/today-im-going-to-talk-about-a-small-practical-example-of-using-neural-networks-training-one-to-6b2cbd6efdb3.

A playful walkthrough of the most basic principles of machine learning.

Project Jupyter Contributors. "JupyterLab," 2021. https://jupyter-cloud.gwdg.de.

An environment for Jupyter Notebooks and Console. The link is to the server hosted by GWDG.

Python Software Foundation. *Python*. V. 3.10.0, October 4, 2020. https://www.python.org/.

An accessible interpreted computer language often used for NLP.

Roberts, Jane, and Christian Kay, eds. *A Thesaurus of Old English*, 2017. With the help of Lynne Grundy, https://oldenglishthesaurus.arts.gla.ac.uk.

A thematically organized index of Old English vocabulary.

Smilkov, Daniel, and Shan Carter. "A Neural Network Playground." https://playground.tensorflow.org/.

A drag-and-drop neural net simulation.

spaCy. V. 3.4. Explosion. https://spacy.io/.

A powerful NLP library for Python which, however, has no support for historical languages.

Stolk, Sander. *Evoke: Exploring Vocabularies and the Concepts Their Words Evoke*. http://evoke.ullet.net.

A frontend for "A Thesaurus of Old English" and other data sets following the same thesaurus structure.

§3: Corpora

Clunies Ross, Margaret, Kari Ellen Gade, Guðrún Nordal, Edith Marold, Diana Whaley, and Tarrin Wills, eds. *Skaldic Poetry of the Scandinavian Middle Ages*. Accessed October 13, 2022. https://skaldic.abdn.ac.uk/.

A current series of print editions of Old Norse poetry, as well as its digital home in the form of a collection of databases.

Healey, Antonette diPaolo, ed. *Dictionary of Old English Web Corpus*. Edited by John Price Wilkin and Xin Xiang. Toronto, 2009. https://tapor.library.utoronto.ca/doecorpus/.

A 3-million-word corpus covering approximately every text (but not every manuscript witness) of Old English. A 2005 release containing HTML and SGML corpora, but not XML, is available from the Oxford Text Archive.

McSparran, Frances, et al., eds. *Corpus of Middle English Prose and Verse*. Accessed October 13, 2022. https://quod.lib.umich.edu/c/cme/.

An XML text corpus of a wide range of print editions.

Orchard, Andy, ed. *CLASP: A Consolidated Library of Anglo-Saxon Poetry*. Accessed October 13, 2022. https://clasp.ell.ox.ac.uk.

An XML corpus of Old English and early Anglo-Latin poetry, searchable by scansion or word type.

Oxford Text Archive. Oxford. https://ota.bodleian.ox.ac.uk/.

A portal providing institutional, and sometimes public, access to a range of digital corpora.

Pęzik, Piotr, Anna Cichosz, Jerzy Gaszewski, and Maciej Grabski, eds. *EnHiGLa*. Accessed October 13, 2022. http://pelcra.pl/enhigla/.

A cross-linguistic corpus of Old High German, Old English, and Latin prose and verse, syntactically annotated.

Pintzuk, Susan, Ann Taylor, Anthony Warner, Leendert Plug, and Frank Beths, eds. *York–Helsinki Parsed Corpus of Old English Poetry*. Accessed October 13, 2022. https://www-users.york.ac.uk/~lang18/pcorpus.html.

Subset of the Helsinki corpus parsed for syntax and parts of speech.

Rissanen, Matti, et al., eds. *Helsinki Corpus of English Texts*, 2011. https://varieng.helsinki.fi/CoRD/corpora/HelsinkiCorpus/.

A diachronic text corpus with minimal markup except for per-text information provided in the TEI header.

Rudolf, Winfried, et al. *Electronic Corpus of Anonymous Homilies in Old English*. Accessed October 13, 2022. https://echoe.uni-goettingen.de.

An Old English XML text corpus comprising all manuscript witnesses of the anonymous and Wulfstanian homilies as well as the prose saints' lives.

Taylor, Ann, Anthony Warner, Susan Pintzuk, and Frank Beths, eds. *The Toronto-Helsinki-Parsed Corpus of Old English Prose*, 2003. https://www-users.york.ac.uk/~lang22/YCOE/YcoeHome.htm.

*A subset of DOEC, parsed for syntax and parts of speech. You can download the actual corpus through the **Oxford Text Archive**.*

§4: Scholarship

"ACL Anthology." https://aclanthology.org/.

An index of journal articles on NLP and computational linguistics.

Adams, Oliver, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. "Cross-Lingual Word Embeddings for Low-Resource Language Modeling." In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, 1:937–947. Association for Computational Linguistics, 2017.

Scales bilingual lexicon-based crosslingual word embedding down to low-resource languages.

Alexander, Marc, and Fraser Dallachy. "Lexis." In *The Routledge Handbook of English Language and Digital Humanities*, edited by Svenja Adolphs and Dawn Knight, 164–184. Abingdon: Routledge, 2020.

An accessible introduction to the concepts of and work in lexical research in English DH.

Bolintineanu, Alexandra. "Wonders that Speak: Computing the Poetics of Wonder in the Old English *Andreas*." *Neophilologus* 106, no. 4, 649–667.

A close reading of passages in the Old English Andreas, *aided by distant reading.*

Caliskan, Aylin. *Detecting and Mitigating Bias in Natural Language Processing*, May 10, 2021. https://www.brookings.edu/research/detecting-and-mitigating-bias-in-natural-language-processing/.

A white paper on the issue of bias in NLP.

Chalmers, David J. "The Puzzle of Conscious Experience." *Scientific American* 273, no. 6 (December 1995): 80–86.

This article has little to do with computing, but gives a sense of the wild west that was the field of philosophy of mind in the decades after Searle's article. I am offering it here just in case the puzzle of consciousness has pique your interest.

Chen, Chi, Maosong Sun, and Yang Liu. “`Mask-Align`: Self-Supervised Neural Word Alignment.” In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 4781–4791. 2021.

Presents a new word aligner with a Transformer architecture.

Christodouloupoulos, Christos, and Mark Steedman. “A Massively Parallel Corpus: The Bible in 100 Languages.” *Language Resources and Evaluation* 49, no. 2 (2015): 375–395.

A project report describing the creation of the corpus, but stopping short of discussing its analysis and applications.

Davis, Matthew Evan, Tamsyn Mahoney-Steel, and Ece Turnator, eds. *Meeting the Medieval in a Digital World*. Leeds: ARC Humanities Press, 2018.

A showcase of recent digital approaches to medieval literature; the contribution by Smith and Butler involves NLP.

Flanagan, Kevin. “Bilingual Phrase-to-Phrase Alignment for Arbitrarily-Small Datasets.” In *Proceedings of the 11th Conference of the Association for Machine Translation in the Americas*, edited by Yaser Al-Onaizan and Michel Simard, vol. 1: *MT Researchers Track*, 83–95. 2014. https://amtaweb.org/proceedings-of-amta-2014/.

Explores dictionary-based approaches to phrase alignment without training or a minimal bitext corpus size, primarily for the purpose of translation memory applications.

Horák, Aleš, and Adam Rambousek. “Lexicography and Natural Language Processing.” In *The Routledge Handbook of Lexicography*, edited by Pedro A. Fuertes-Olivera, 179–196. London: Routledge, 2017.

A brief overview of the value of each of these fields for the other.

Huang, Po-Sen, Huan Zhang, Ray Jiang, Robert Stanforth, Johannes Welbl, Jack W. Rae, Vishal Maini, Dani Yogatama, and Pushmeet Kohli. “Reducing Sentiment Bias in Language Models via Counterfactual Evaluation.” *Findings of the Association for Computational Linguistics*, 2020, 65–83.

Research into bias reduction in sentiment analysis.

Jänicke, Stefan, and David Joseph Wrisley. “Interactive Visual Alignment of Medieval Text Versions.” In *2017 IEEE Conference on Visual Analytics Science and Technology*, edited by Brian Fisher, Shixia Liu, and Tobias Schreck, 127–138. Piscataway, NJ: Institute of Electrical / Electronics Engineers, 2017. https://doi.org/10.1109/VAST.2017.8585505.

Report on a tool allowing computational text alignment to be adjusted by the researcher.

Lopez, Adam. “Statistical Machine Translation.” *AC Computing Surveys* 40, no. 3 (August 2008): article 8.

A detailed survey article on statistical machine translation.

Meinecke, Christofer, and Stefan Jänicke. “Automated Alignment of Medieval Text Versions based on Word Embeddings.” In *LEVIA 2019: Leipzig Symposium on Visualization in Applications*. 2019. https://doi.org/10.31219/osf.io/tah3y.

An effort to bring word embeddings to bear on text alignment alongside conventional methods.

Reinke, Uwe. “State of the Art in Translation Memory Technology.” In *Language Technologies for a Multilingual Europe: TC3 III*, edited by Georg Rehm, Felix Sasaki, Daniel Stein, and Andreas Witt, 55–84. Translation and Natural Language Processing 5. Berlin: Language Science Press, 2018. https://langsci-press.org/catalog/book/106.

An accessible introduction to translation memory and its points of contact with machine translation.

Ruder, Sebastian, Ivan Vulić, and Anders Søgaard. “A Survey of Cross-Lingual Word Embedding Models.” *Journal of Artificial Intelligence Research* 65 (2019): 569–631. https://www.jair.org/index.php/jair/article/view/11640/26511.

Searle, John R. “Minds, Brains, and Programs.” *The Behavioral and Brain Sciences* 1980 (3 1980): 417–457.

Puts forth the foundational Chinese room argument on the meaning of understanding in the context of computing.

Shi, Haoyue, Luke Zettlemoyer, and Sida I. Wang. "Bilingual Lexicon Induction via Unsupervised Bitext Construction and Word Alignment." In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 813–826. Association for Computational Linguistics, 2021.

A highly technical article presenting an improved method for the creation of bilingual lexicons on the basis of bitexts.

Smith, Samuel L., David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. "Offline Bilingual Word Vectors, Orthogonal Transformations and the Inverted Softmax." In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings*. https://openreview.net/forum?id=r1Aab85gg.

Introduces a technique for aligning two or more monolingual sets of word embeddings with an orthogonal constraint.

Smith, William H., and Charles L. Butler. "Statistical Analysis and the Boundaries of the Genre of Old English Prayer." In Davis, Mahoney-Steel, and Turnator, *Meeting the Medieval in a Digital World*, 11–26.

Applies relative term frequency to compare Old English prayers.

Sprugnoli, Rachele, Marco Passarotti, Daniela Corbetta, and Andrea Peverelli. "*Odi et amo*: Creating, Evaluating and Extending Sentiment Lexicons for Latin." In *Proceedings of the 12th Language Resource and Evaluation Conference*, 3078–3086. Marseille: European Language Resources Association, May 2020. https://aclanthology.org/2020.lrec-1.376/.

Describes efforts to create Latin sentiment lexicons for use in sentiment analysis.

Stolk, Sander, and Thijs Porck, eds. "Exploring Early Medieval English Eloquence: A Digital Humanities Approach with 'A Thesaurus of Old English' and Evoke." *Amsterdamer Beiträge zur älteren Germanistik* 81.

A special issue on the Evoke application and its Old English data sets.

Tiedemann, Jörg. *Bitext Alignment*. Synthesis Lectures on Human Language Technologies 14. Springer, 2011.

An accessible introduction to bitext preparation and alignment on all levels.

Torabi, Katayoun. "If (Not "Quantize, Click, and Conclude"): {Digital Methods in Medieval Studies}." In Davis, Mahoney-Steel, and Turnator, *Meeting the Medieval in a Digital World*, 27–44.

A survey arguing that available tools do not replace close reading.

Vaswani, Ashish, et al. "Attention Is All You Need." In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, edited by Ulrike von Luxburg et al., 5999–6009. 10 vols. Advances in Neural Information Processing Systems 30. La Jolla, CA: Neural Information Processing Systems, 2017.

A highly influential but for our purposes somewhat technical article on bitext processing.

Véronis, Jean, ed. *Parallel Text Processing: Alignment and Use of Translation Corpora*. Dordrecht: Kluwer, 2000.

A volume of essays on methods and applications of bitexts. Out of print.

Wunderlich, Martin, Alexander Fraser, and P. S. Langeslag. "'God wat þæt ic eom god': An Exploratory Investigation into Word Sense Disambiguation in Old English." In *Proceedings of GSCL 2015: International Conference of the German Society for Computational Linguistics and Language Technology*, 39–48. Munich: Gesellschaft für Sprachtechnologie und Computerlinguistik, 2015. https://konvens.org/proceedings/2015/.

Tests a selection of supervised approaches for sense disambiguation for Old English.

Younes, Nadja, and Ulf-Dietrich Reips. "Guideline for Improving the Reliability of Google Ngram Studies: Evidence from Religious Terms." *PLoS ONE* 14 (3 2019). https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0213554.

Zhang, Sarah. "The Pitfalls of Using Google Ngram to Study Language." *Wired*, October 12, 2015. https://www.wired.com/2015/10/pitfalls-of-studying-language-with-google-ngram/.