The *Lexos* Experience and the Future of Accessible Text Analysis Tools

Introduction

Lexos is a web-based text analysis tool which offers an integrated workflow for users to perform pre-processing of textual data, along with an array of methods of analysis and visualisation tools, primarily focused on studying the similarity and distinctiveness of single-token vectors. *Lexos* offers an interface designed to be used by novices, with plenty of built-in help documentation. It is ideal for classroom use but powerful enough to be used for scholarly research. The tool is written in Python and JavaScript, and the open-source code is available on GitHub and can be downloaded and installed locally. However, the online instance hosted by Wheaton College, in Massachusetts is used widely for teaching, especially in introductory DH classes.

Lexos got its start in 2008 as a curricular experiment at Wheaton. With a bit of seed funding from the National Endowment for the Humanities, Professors Mark LeBlanc, a computer scientist, Michael Drout, a specialist in medieval English literature, and Mike Kahn, a statistician, were brought together to design an interdisciplinary course that brought their various areas of interest into dialogue. With undergraduate participation, they began to explore techniques for computational literary analysis, which they dubbed ‘Lexomics’, focussing initially on the use of hierarchical cluster analysis for the study of Old English literature. The team employed PERL scripts that performed pre-processing on the corpus assembled by the *Dictionary of Old English* and called clustering routines in R to produce dendrograms showing textual similarity. I became involved in the project in 2011 by developing a web-based interface for the project’s scripts, and I have had a lead role in the project ever since. Between 2011 and 2017, *Lexos* was funded by two further grants from the NEH.

Development was also supported by Wheaton’s summer research fellowships, which brought teams of undergraduate English and Computer Science majors together in a lab each summer until the beginning of the pandemic. The English students would use *Lexos* to engage in new research whilst the Computer Science majors would develop the web application in response to the needs of the student and faculty research agenda. As time went on, the complexity of *Lexos* required its developers to pay more and more attention to developing a robust and sustainable coding infrastructure that followed standards and exemplified best practices in software development. *Lexos* is now in version 4.0 and has changed considerably over the years in response to changing technologies and the expertise of Wheaton’s undergraduate developers. This unique model has been an undeniable benefit to its student participants, who have authored publications and gained valuable experience in software engineering. I will discuss other implications of the model for the development of *Lexos* below.

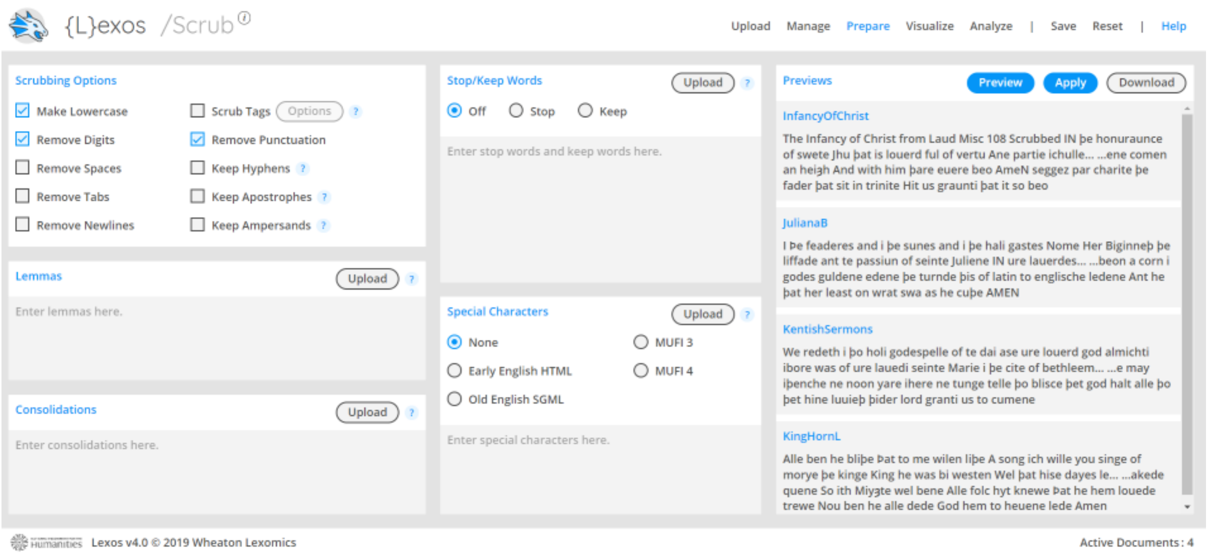
As *Lexos* has evolved, we have tried to remain faithful to certain fundamental principles. *Lexos* was designed to lower the barrier to entry to computational text analysis. It is ideal both for use as a teaching tool (suitable for the undergraduate classroom) and as a research tool for any researcher who might lack coding expertise. The interface is designed to overcome the opacity of computational methods by integrating help features that open up the algorithmic 'black box' – although the precise means of achieving this goal have changed over the years. We have also made it a priority to include features that help students and scholars working with pre-modern or under-resourced languages overcome the types of problems we encountered in studying Old English in the early days of the project. This means that we have put considerable effort into developing *Lexos*' pre-processing tools to allow users to perform rich manipulations to make their data tractable for computational analysis.

Lexos is used widely in introductory Digital Humanities courses, although we have had a difficult time collecting data about its user base. *Lexos* has obvious similarities to Voyant Tools, particularly in having an accessible web-based interface. It also resembles Stylo in R in that it heavily emphasises hierarchical cluster analysis (the method originally investigated by the Lexomics group). In technical terms, it differs from both in employing Python and its available statistical libraries on the back end. Readers are encouraged to try it out to get a sense of the range of features offered in *Lexos*. Test data and guidance are available in the Experiments folder in the *Lexos* GitHub repo. However, the short overview should suffice as context for my discussion of how *Lexos* can continue to support users and how it can evolve in the changing Digital Humanities landscape.

Overview of Features

Users typically upload their data as plain text files, although users can also supply .docx files, tagged HTML and XML files, and URLs. In all cases, the materials are downloaded to the server (or a local session directory for local installations) and coerced into UTF-8 encoding. Once files are downloaded, users can use the Manage tool to activate and de-activate documents in their corpus. Inactive documents are ignored by the other *Lexos* tools but can be made active at any time. Although *Lexos* should be able to handle large numbers of files, its UI is not designed for ingesting or managing the large datasets that are increasingly available to and used by members of the DH community. We conceive of *Lexos* as best suited to small to medium-sized datasets.

The usual first step is to perform pre-processing functions, known in *Lexos* parlance as 'scrubbing'. The Scrub tool (AKA 'Scrubber') allows the user to select common functions such as removing digits, punctuation, or stop words, consolidating string patterns (important for languages with multiple equivalent spellings), stripping markup tags, and decoding entities in various markup languages. A variety of special character entities commonly used in the markup of early European languages are available out of the box, but entity sets can be uploaded or entered manually.


{L}exos /Scrub
Upload Manage Prepare Visualize Analyze | Save Reset | Help
Upload
Preview Apply Download
Lexos v4.0 © 2019 Wheaton Lexomics

Scrubbing Options
- [✓] Make Lowercase
- [✓] Remove Digits
- [ ] Remove Spaces
- [ ] Remove Tabs
- [ ] Remove Newlines
- [ ] Scrub Tags (Options)
- [✓] Remove Punctuation
- [ ] Keep Hyphens
- [ ] Keep Apostrophes
- [ ] Keep Ampersands

Lemmas: Enter lemmas here.

Consolidations: Enter consolidations here.

Stop/Keep Words: [✓ Off] Stop Keep — Enter stop words and keep words here.

Special Characters: [✓ None] Early English HTML, Old English SGML, MUFI 3, MUFI 4 — Enter special characters here.

Previews

InfancyOfChrist
The Infancy of Christ from Laud Misc 108 Scrubbed IN þe honuraunce of swete Jhu þat is louerd ful of vertu Ane partie ichulle... ...ene comen an heiȝh And with him þare euere beo AmeN seggez par charite þe fader þat sit in trinite Hit us graunti þat it so beo

JulianaB
I Þe feaderes and i þe sunes and i þe hali gastes Nome Her Biginneþ þe liffade ant te passiun of seinte Juliene IN ure lauerdes... ...beon a corn i godes guldene edene þe turnde þis of latin to englische ledene Ant he þat her least on wrat swa as he cuþe AMEN

KentishSermons
We redeth i þo holi godespelle of te dai ase ure louerd god almichti ibore was of ure lauedi seinte Marie i þe cite of bethleem... ...e may iþenche ne noon yare ihere ne tunge telle þo blisce þet god halt alle þo þet hine luuieþ þider lord granti us to cumene

KingHornL
Alle ben he bliþe Þat to me wilen liþe A song ich wille you singe of morye þe kinge King he was bi westen Wel þat hise dayes le... ...akede quene So ith Miyȝte wel bene Alle folc hyt knewe Þat he hem louede trewe Nou ben he alle dede God hem to heuene lede Amen

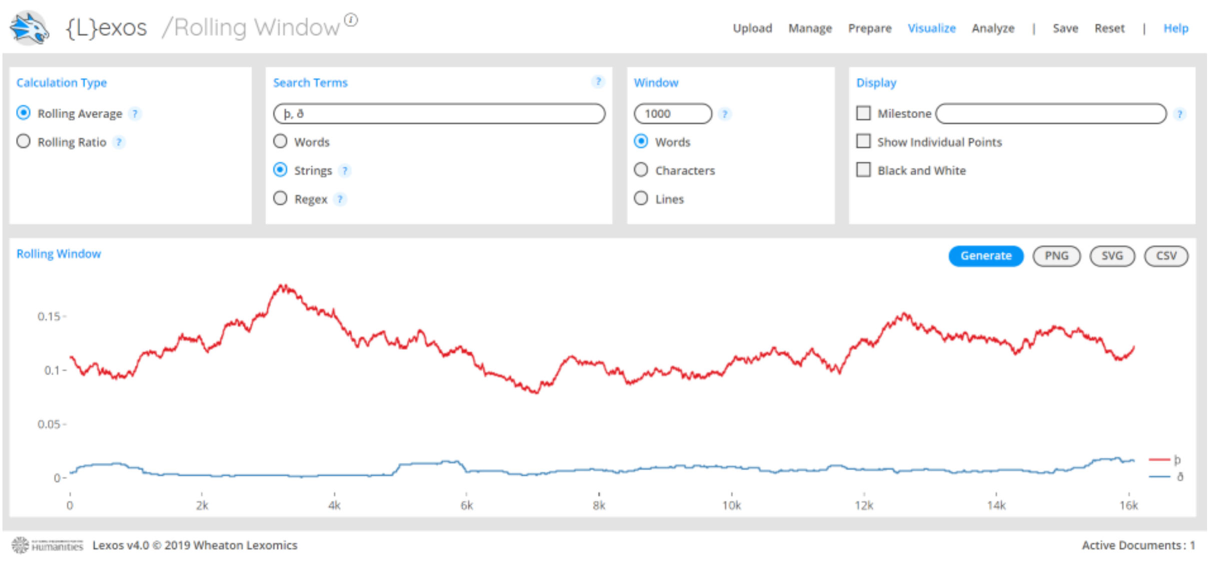
Active Documents: 4

Screenshot of the Lexos Scrubber Tool

Users can also use the Cut tool (AKA 'Cutter') to split their documents into segments based on word or character length, as well as on structural markers like chapter divisions. They can also use the Tokenize tool to generate a table of document-term counts or frequencies, which can be culled to include the most or least frequent terms. All these tools allow the user to preview the results without modifying the original documents, and the results can be downloaded. In some cases, the user's only goal may be to obtain a table of token counts.

One important caveat is that these tools operate by matching character patterns to obtain countable tokens. Whilst it is often possible to approximate a 'word' token by removing punctuation and splitting the remaining text on whitespace, this does not work in all languages. *Lexos* provides some tools for the user to manipulate texts in languages like Chinese, which do not divide words with spaces, to obtain meaningful results. However, *Lexos* does not employ any pre-existing knowledge of the document's language. This is a subject I will return to below.

Once users have prepared their documents to their satisfaction, they can proceed to a number of analysis and visualisation tools. Wordcloud, Multicloud, and BubbleViz are all tools users can employ to visualise the document-term matrix in slightly different ways. The most distinctive tool is Rolling Window, which enables the user to plot a line graph showing the average frequency of pattern (or ratio between two patterns) in a sliding window of text of a desired size.



Calculation Type: [✓ Rolling Average] Rolling Ratio

Search Terms: þ, ð — Words [✓ Strings] Regex

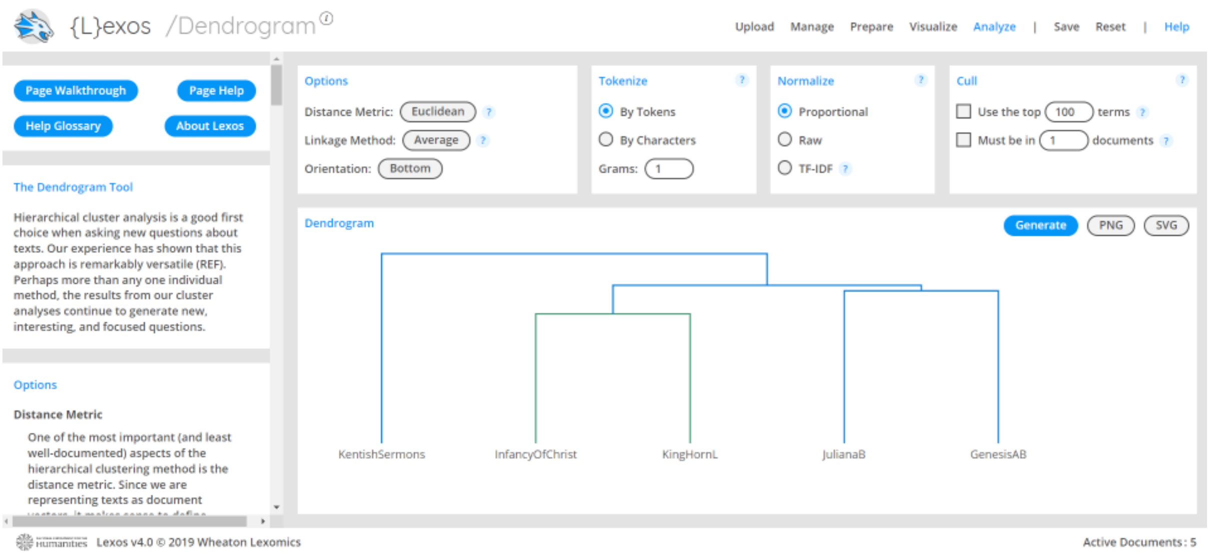
Window: 1000 — [✓ Words] Characters Lines

Display: ☐ Milestone ☐ Show Individual Points ☐ Black and White

Lexos v4.0 © 2019 Wheaton Lexomics — Active Documents: 1

Screenshot of the Lexos Rolling Window tool

Lexos is primarily used for its Dendrogram tool, which performs hierarchical cluster analysis. The interface allows the user to switch between common distance metrics and linkage methods, as well as to tokenise the data in the various ways performed by the Tokenize tool. The Help sidebar provides guidance on how to choose amongst the various options.



The Dendrogram Tool

Hierarchical cluster analysis is a good first choice when asking new questions about texts. Our experience has shown that this approach is remarkably versatile (REF). Perhaps more than any one individual method, the results from our cluster analyses continue to generate new, interesting, and focused questions.

Options

Distance Metric

One of the most important (and least well-documented) aspects of the hierarchical clustering method is the distance metric. Since we are representing texts as document

Options: Distance Metric: Euclidean; Linkage Method: Average; Orientation: Bottom

Tokenize: [✓ By Tokens] By Characters; Grams: 1

Normalize: [✓ Proportional] Raw TF-IDF

Cull: ☐ Use the top 100 terms; ☐ Must be in 1 documents

Lexos v4.0 © 2019 Wheaton Lexomics — Active Documents: 5

Screenshot Lexos of the Lexos Dendrogram tool with the Help sidebar open

For comparison, users can also use the K-Means tool to perform k-means clustering, the Similarity Query tool to calculate cosine similarity between documents and a host of metrics for assessing the distinctiveness of terms and documents within the corpus.

The web browser will usually remember a user's session if they leave the site, although sessions are periodically deleted on the hosted version. However, users can download a zip file of their workspace and then upload it to resume work where they left off.

Challenges to Further Development

Lexos has many challenges if it is going to be a viable tool for DH teaching and research in the future. This is the depressing section of my discussion, so I want to address these challenges now before turning to the more optimistic subject of what *Lexos* and other text analysis tools might look like if they are positioned to serve DH users in the future.

Sustainability

One of the primary challenges is the institutional environment in which *Lexos* has developed. At its inception and for many years, we benefitted from a rich supply of talented developers in the Wheaton undergraduate student body. However, since the pandemic, Wheaton's funding has begun to dry up, making it difficult to recruit students who do not require a considerable amount of training. As the code base has grown, so has the amount of training required. Furthermore, Mark LeBlanc, the Wheaton computer scientist who has taken the lead in these efforts, is nearing retirement and has begun scaling back his commitment to the project. I have taken the lead on the future development of *Lexos*; however, as a professor at a non-selective state-funded institution with a heavy teaching load, I have little institutional support for my labour and cannot draw on the same student talent pool for the types of activities that have spurred development previously. At present, I am not even sure that I will be able to provide an institutional host for the public instance of *Lexos* once Professor LeBlanc retires. Managing the infrastructure and a continuous flow of student developers – not to mention the 'marketing' that is required for a tool to gain traction within the DH community – is no mean feat even at more well-resourced institutions, but this represents a particular challenge for *Lexos*.

That so many students had the opportunity to take important roles in software development was a huge selling point at the beginning of the project but it has also had some unfortunate effects. For instance, in 2020 the students made extensive changes to both the back-end architecture and front-end interface. There was a rush to complete the rewrite so that graduating seniors could put a line on their CVs. As a result, certain features available in the previous release were not implemented in the new version, and a degree of usability was lost in the new design. Professor LeBlanc and I found ourselves no longer familiar with the code base, which has hampered further development. Although we continue to make tweaks, and *Lexos* 4.0 is running on the public server, we have not yet made it an official release.

Changing Landscape

A related challenge that concerns this panel is how the needs of our audience change as the DH landscape evolves. For some time, the various strands of computational text analysis (stylometry, computational literary studies, cultural analytics, distant reading) have been increasingly moving beyond (single) vectors of word counts towards more complex word embeddings, NLP pipelines, and even LLMs. These changes present designers of existing tools like *Lexos* with several questions. Is there value in maintaining an older tool offering older techniques? If so, can new techniques be implemented within the existing technical structure? Can they be implemented within the existing user interface?

These are big questions to which I will provide only brief answers deriving from the *Lexos* experience. Many new techniques require large datasets which simply aren't available for some humanities subject matters. There thus remains a need for tools that specialise in the analysis of smaller datasets. Similarly, tools which build in solutions for particular problems like non-standardised spelling or provide clear documentation about how particular algorithms work are vital to making computational text analysis accessible to a wider audience that may lack the technical expertise or the time to design their own solutions. Until recently, a well-designed user interface that you could access on the web or launch with a double click was valuable to this audience. Notebook coding environments have begun to erode the advantage of such interfaces by combining text-based explanations with implementable code. Increasingly, I am seeing dashboard-like mini-apps being produced using tools like the Python Streamlit library which allow users to perform specific manipulations of data. The next stage of evolution is likely to be the AI-generated interface, produced quickly and for a similar limited purpose. (I will consider the prospect of generative AI chatbots as interfaces below.) For now, I will simply say that we may have to reconceive what constitutes an 'interface' for a text analysis tool in the future. In the next section, I will describe the direction I have taken *Lexos* in.

Re-Inventing *Lexos*

Two years ago, I attempted to restore some of the features that were lost after the Summer of 2020, but I found the code difficult to understand and concluded that it was likely to be impossible for future developers to create new features without a substantial learning curve. As a result, I decided to re-implement all the features as a separate Python library that would be documented obsessively using modern documentation tools. I hope that making *Lexos* functions independent of the web app will make it easier to develop new features without the overhead of designing a front end around them. Eventually, the front end that calls the Python code via simple API requests should be considerably simpler to maintain, and, in the case of *Lexos*, adapt to changes in web technologies. The pre-release of the `lexos` Python library is now available on GitHub, although I emphasise that it is very much not ready for primetime.

Besides making a more efficient and sustainable developer experience, I began to re-imagine the architecture of *Lexos* by fundamentally changing its basic units of analysis. The web app currently stores document files as plain text strings, which it divides into tokens based on regular expression patterns, normally splitting on whitespace. *Lexos* leaves it up to the user to tweak the default tokenisation as necessary to approximate the language or other requirements of their source material. The new `lexos` library instead uses existing language models available in the latest NLP technologies to implement tokenisation. The token as a unit is no longer a simple string; it contains a set of annotations or 'attributes' which provide further information about its semantic content, including features such as lemma, part of speech, morphological form, named entity type, or other bespoke information. Users can now access this information to interrogate a more diverse range of features beyond the simple frequency of word vectors (where 'word' really means a type of character string). This helps to address one of the major criticisms of early word vector-based analysis, which is that it was largely context-independent. By enabling users to access some of the latest NLP technologies, it also expands the number and type of research questions that can be investigated based on the annotations generated using language models.

Without getting into the technical weeds, I will say that the vehicle for this transformation is the Python NLP library spaCy, which is fairly well-known in the DH community. The `lexos` library encourages users to convert their texts to spaCy Doc objects as soon as possible and then uses spaCy's built-in methods to perform tokenisation and access token attributes. The advantage is that spaCy has an ever-growing set of language models which allows the *Lexos* user to generate token attributes appropriate to the language of their texts. (*Lexos* uses a default model if no spaCy or spaCy-compatible model is available.)

These choices come with consequences. One is that work on the user interface, which has been so vital to making *Lexos* accessible, is for the moment on hold. At the same time, the existence of a well-documented Python library (which can be installed with `pip install lexos` command) makes it possible for users with a little coding knowledge to implement the features of *Lexos* fairly easily. In this sense, it moves *Lexos* in the same direction as Spyral, which allows users to deploy Voyant Tools in a notebook-like environment. This direction arguably shifts the audience for *Lexos* away from the introductory DH student or the non-coding Humanities scholar towards users with a pre-existing degree of technical knowledge. Part of me recoils at this development, as I still think that the menus, check boxes, and other settings of the web-based user interface are important for reaching these audiences. Another part of me shies away from being drawn into the old debate about where digital humanists, or students, generally need to learn to code. But it may be a reality that text analysis tools like the *Lexos* web app can only provide a 'gateway' to a limited set of features, and more sophisticated operations need to take place in other environments. One simple example must suffice to exemplify this point. SpaCy tokens may be tagged with over 30 different attributes and possess the ability to be annotated by the user with their own custom attributes. The functions in the `lexos` Python library can access these attributes with keywords, but designing an interface to give users access to all of them from the front end is a tall order. Even if it could be done, users might prefer to configure the functions themselves in a coding environment. This comes with its own danger. *Lexos* must be more than a thin wrapper around spaCy (or scipy or plotly, or any of the other Python libraries of which the *Lexos* web app already makes heavy use). It must provide some value added to give the user some reason to use it, as opposed to interacting with those libraries directly.

My partial answer to these dilemmas is that *Lexos*, whether or not it has a user interface, has to be designed with ready-made solutions to Digital Humanities problems: that is, problems faced by students and scholars who are primarily interested in applying computer technologies to humanistic questions and content. This may take the form of code routines that script particular workflows or discussion embedded in the documentation that defines terms or describes best practices for Humanities audiences and data. For this reason, I am trying to supplement documentation of the `lexos` API aimed at developers with tutorials that introduce concepts and workflows for the types of audiences served by the *Lexos* web app.

There are a few other points to be made about the re-invention of *Lexos* as a Python library. I hope that this process will make the future of *Lexos* less reliant on ever-changing teams of student developers, or, indeed, a sole developer at a non-research institution with little. Instead, I hope to make *Lexos* more like a typical open-source software project. Once I reach the beta stage, I will invite members of the DH community to contribute to the code base, and I hope that there will be sufficient interest to generate enhancements based on the needs of DH users. In advance of this, I am already soliciting suggestions about what users would like to see in *Lexos*. I have already implemented one such suggestion for the Rolling Window tool, which will be integrated into the next version of the `lexos` Python library.

The separation of *Lexos*' functionality from its interface is also leading me to explore the use of *Lexos* to develop tools for other digital projects. For instance, I am collaborating with the New Variorum Shakespeare Project to create a *Lexos* visualisation plugin for their platform. I am interested in exploring other collaborations and am keen to discuss with the producers of other text analysis tools ways in which data could be passed easily from one tool to another. I like the idea of text analysis tools being in dialogue, and I will have more to say about that below.

What Should a DH Text Analysis Tool Look Like in the Future

In this final section, I wish to speculate about what *Lexos* and other text analysis tools might ideally look like if we could set aside the types of challenges I have described.

Audience

An out-of-the-box text analysis tool can have multiple audiences from introductory students to advanced scholars, and it can be a useful timesaver even for those who might be able to implement its functions programmatically. However, there are challenges to figuring out who is using the tool, for what purposes, and how the tool can continue to be relevant as new technologies and methods are adopted by digital humanists. A tool cannot be all things to all people, but some consideration should be given to how it can be designed to accommodate multiple levels of expertise. A flexible audience may create more of a challenge for developers, but it may help to expand the user base and encourage design that fosters the adoption of new methods and technologies. However, it is important to retain accessibility as a goal for every level of complexity.

Ease of Use and Accessibility

I believe that an audience that might be described generally as 'low-resourced' is best served by a tool that is designed with user experience in mind. Based on my experience with *Lexos*, I would make the following recommendations:

- Tools aimed at an entry-level audience should have an interface available in a hosted environment on the web. If the web browser or web hosting provides limitations, the application should at least be easy to install locally.
- The user interface must be aimed at helping users in the Humanities operationalise the types of procedures they are likely to want to implement with their materials such as manipulating non-standardised forms of language or TEI-encoded texts (more on the latter below).
- The user interface should provide – or at least point to – documentation with easy-to-understand explanations of the algorithms used, their parameters, and best practices for deploying them for use with different types of Humanities data.
- The user interface must serve to bridge the epistemological and methodological knowledge gaps that divide the disciplines, not just enable push-button implementation of algorithms that can be performed programmatically.
- If the tool requires a coding environment, like the `lexos` Python library, it is still important to employ these principles.

Community and Support

Another principle that has emerged from the *Lexos* experience is the importance of community, both for sustainability and continued relevance. In the early years of the project, Lexos benefitted from THATCamp and other workshop-style gatherings which are less common these days. From a developer's point of view, following best practices for the design of open-source software is, in my view, essential. I have often wondered whether the developers of major text analysis tools should have a website, Discord server, GitHub repo, or another place to come together to talk through how their efforts overlap, dovetail, or otherwise relate. For instance, does *Lexos* have to implement a particular type of word vector visualisation if it already exists in Voyant Tools? Would it not be more useful to establish a procedure to perform manipulations in *Lexos* and then pipe the results into Voyant Tools? Another advantage of having an open forum to engage in dialogue around off-the-shelf text analysis tools is that developers would be in a better position to understand their audiences and what features those audiences most require.

Thinking about the place of tools like *Lexos* in a community can also help them play an important function in helping to unify the Digital Humanities as a field. Let me illustrate this by discussing a feature of *Lexos* that was broken during the summer of 2020. This was the ability to perform fine-tuned scrubbing of XML, and thus TEI-XML, tags. If you are studying the text of a digital edition, the practice of stripping all tags from the text before performing computational textual analysis – particularly of the bag of words type – ironically undoes all the hard work of our colleagues in the branch of DH who are dedicated to enriching digital texts with semantic markup. As more and more textual content becomes available with descriptive markup, we do a disservice to the field if we design tools that can not make use of it. In turn, the designers of digital editing and archiving projects – like the New Variorum Shakespeare Project – should increasingly have in mind making their content available to text analysis tools. TEI markup is analogous to the token attributes produced with NLP tools like spaCy, and we should be able to leverage that extra information they provide when we perform computational text analysis. I think that the DH tools of the future, particularly those aimed at entry-level users, should attempt to bring the work of these two communities more closely together. It may not answer the perennial 'What is DH?' question, but it may help people gain a better understanding of the broad parameters of our field and enable them to find an accessible entry point into further DH study.

In short, future text analysis tools could

- Form part of a coherent ecosystem in which tools with different functions can talk to each other.

- Be located within a community forum where dialogue takes place about their use to address common problems in the Digital Humanities.
- Play an important role in bringing together diverse strands of the Digital Humanities.

Sophistication

Anyone who has ever taught TEI knows that it has a tremendous learning curve, especially for student learners, who must master the application of a complex schema to a particular type of material and set of editorial priorities. The same is true for text analysis, and the complexity of the computational methods used by advanced researchers has increased tremendously in recent years. We should ask whether off-the-shelf DH tools for which students are the primary audience should implement these methods. I would argue yes because not doing so risks obsolescence and irrelevance. We want to be able to teach students to ask questions and study data in ways that are enabled by current technologies like language models. However, we do need to be realistic about the limitations such as the size of data required or the costs of implementation on high-performance servers. In the coming years, cost and scalability may present insurmountable barriers to the implication of sophisticated algorithms in entry-level tools.

I think it is also important for scholars in the Humanities to use the tools to perform real research. In my view, if a tool merely functions as an 'intro to DH', its value – and thus the value of the methods it demonstrates – becomes too theoretical. It is better for students to be able to see what their professors are doing in their research and maybe even to use the tool to participate in that research.

Furthermore, I think that many time-strapped researchers whose primary focus is on humanistic questions need easily implemented versions of the latest methods, as well as resources like language models that already solve, or partially solve, problems like how to deal with orthographic or scribal variation in pre-modern writing systems. Sometimes that calls for tools like an interface for stripping critical markup from diplomatic markup without the need to use an XML parsing library or (oh the horror!) XSLT. I have no better advice than to say that designers of off-the-shelf text analysis tools may need to look for opportunities to provide access to aspects of data- or computationally-intensive methods, but recognise that their strengths are more likely to be in solving higher-level workflow problems. The best medium for bringing wide audiences to text analysis using tools like LLMs may be a smaller mini-app, rather than a fuller, multi-featured application.

This brings me to consider, at last, the emerging role that AI might have in Digital Humanities text analysis. Ziems, et al. have recently argued that the ability of AI to ‘retrieve, label, and condense relevant information at scale’ is promising, at least in some domains, and that by ‘labeling human sample flexibly in low-cost classification capabilities’, it may even lead to the creation of new research paradigms. I consider this very promising indeed. Already, the *Lexos* Manage tool allows users to provide class labels for their documents, but it might be useful to allow an AI to suggest class labels. This is a relatively inexpensive operation, and the user might make cluster analyses with different sets of labels to see how the results compare. Ziems, et al. suggest AIs can help generate annotations to train language models (albeit, with humans still in the loop). That would potentially lower the cost of enriching the data available for analysis, enhancing the types of research which I hope the new `lexos` Python library will enable. In addition to tools for analysis, we may need to begin to devote attention to developing entry-level tools for training models for Humanities analyses. We may need to produce and host our own models, rather than large general-purpose models trained on large datasets scraped from the web. Again, having some kind of community platform might help us share resources.

I am less certain how generative API chatbots, trendy as they are, might be deployed in off-the-shelf text analysis tools. Perhaps a built-in chatbot might be used to explain features of the interface or help a user decide whether to use Euclidean distance or cosine similarity on their dataset. A chatbot that uses a model trained specifically on DH publications might provide genuinely useful results (especially if it is also trained to provide an appropriate *caveat emptor*). In the future, generative AI may help us to create truly paradigm-shifting forms of analysis based on the features that transformer-based LLMs apparently extract from subword tokens. But since research into what exactly is happening inside LLMs is just beginning, I think speculation about its significance is perhaps best left for a follow-up discussion, perhaps at DH2034. Who knows? By then we may have some non-human participants on the panel.