Word Vectors and Text Analysis

This workshop introduces the Women Writers Vector Toolkit, an interface that allows researchers to conduct word vector analyses on texts already available in the Women Writers Project, a digital collection of works authored by women from 1400 to 1850. This introductory workshop will use open-access web tools hosted in the Women Writers Online Lab since 2013.
https://wwp.northeastern.edu/lab/wwvt

Introduction

As humanists, we will explore word embedding models, a group of text analysis. Word vectors offer an opportunity to explore the semantic spaces and relationships within a large corpus, discover analogies between words, and study details of register and genre. We will also consider how we can make and assess arguments about and with text analysis data, and discuss how to evaluate the validity of methods, data preparation, and tool configuration.

Word vectors operate in an environment of word embedding models. This type of text analysis explores larger thematic questions in extensive collections of texts. Computational tools display spatial analogies: their methods represent and visualize relationships between words.

Terminology

Let us define some key terms and concepts:

- Corpus:
    - a collection of documents
    - the body of textual material we are analyzing
    - a set of documents in some machine-readable form: some cleaning and regularization might be necessary, so that we have the plain-text corpus that is going to be fed into Word2Vec tool

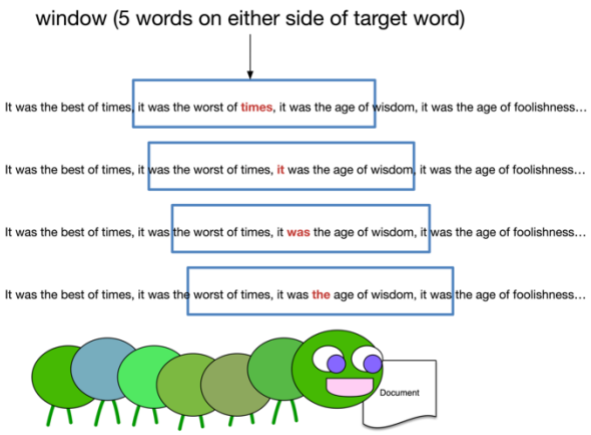
- Model:
    - a representation of something we are interested in: it shows some, not all the important aspects, so that we can study and have an overall understanding of the item(s)
    - a representation of a collection of texts (a corpus), enhancing the semantic relationships between words, so that we can see and study them
    - a processed representation of the textual data contained in those documents
    - a computed version of the corpus, produced by the Word2Vec tool, representing the semantic positioning of each word in the corpus as a vector
    - creating a model is not a one-step action: more attempts take us as close to a "perfect" model, as possible

- Training process
  - how we get from the corpus to the model
  - the process resembles the production of an item from scratch, to a finished product. Once you have the ingredients, you pour them in a big machine, and you get out some product: a bag of cookies, for example
  - the time to train is proportional to what categories you include: on a 16-GB laptop, for a good-sized corpus, it is about one hour per iteration

- Parameters
  - the settings in the machine, so that we can have a good final product

- Window
  - Assumptions might be there: words that are used together have something to do with one another
    - right next to one another?
    - relevant because they are closer to one another?
  - In word vectors analysis, a window is a span of text of a specified length
  - We can control the size of the window by setting parameters
    - a bigger window lets us treat larger groups of words as related
    - what might be the results for our analysis of a larger or smaller window? (imagine a window that is an entire chapter; imagine a window that is only two words wide)
  - In the analogy developed by WWP at Northeastern, the Word2Vec algorithm is "like a caterpillar eating its way through the text, bite by bite. Each taste is localized by the window: more local if the window is smaller, less local if the window is larger . . . each bite gives the processor a set of words that are considered used together. And the size of the bite affects what is considered proximate to what"

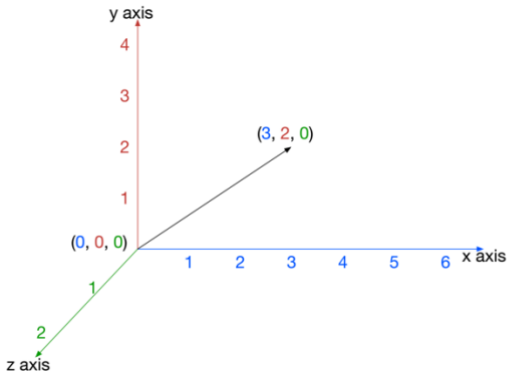
- Iterations
  - Training means practice. Thus, several attempts ("iterations") are necessary. Every time you repeat the machine reading, one small adjustment helps us to achieve a better picture of the model
  - The first time, the process makes a set of observations, and so on, until we have a more accurate model
  - Since we are working with textual sources, there is a constant re-reading of the plain text materials – as we also usually read one source more than once, in order to understand and study it fully
  - Not only the model becomes more accurate with more iterations… we also need more time. However, we can control the number of iterations as a parameter setting



Here we see a visual metaphor for windows and iterations (https://www.wwp.northeastern.edu)

Let's Get Started

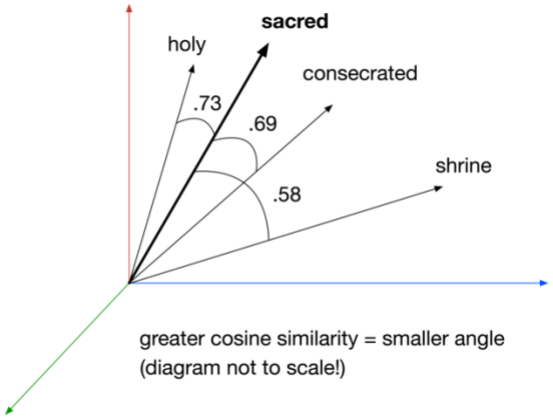
We are using a web tool for word vectors and text analysis, without having to do the programming work. The Women Writers Vector Toolkit allows us to interact with the trained models, that have become standardized collections. The toolkit and its exploratory interface combine a set of programs for Word2Vec, via RStudio: https://wwp.northeastern.edu/lab/wwvt.



A vector is a line that has both a specific length and a specific direction or orientation in space. In a word-embedding model, the model represents a text corpus so that, in a certain sense, each word projects some meaning, based on its position and proximity in vector space.

Vectors and vector models are important to understand facets of a collection: not only the presence of certain words, but also the absence is something that is important to note. As a matter of fact, proximity (semantically) translates to proximity (spatially).

Visualizations and Mathematical Aspects



Cosine similarity measures the nearness among words. It is a geometrical measure of the angle between two vectors, with values ranging between zero and one.

- Two identical vectors have a cosine similarity of 1
- Two absolutely dissimilar vectors have a cosine similarity of zero
- The smaller the cosine similarity, the smaller the similarity between the words
- Above .5? good to consider

Let us try to understand the connections in the above graph: starting from "sacred," how close are the words "holy," "consecrated," and "shrine"?

Querying a Model

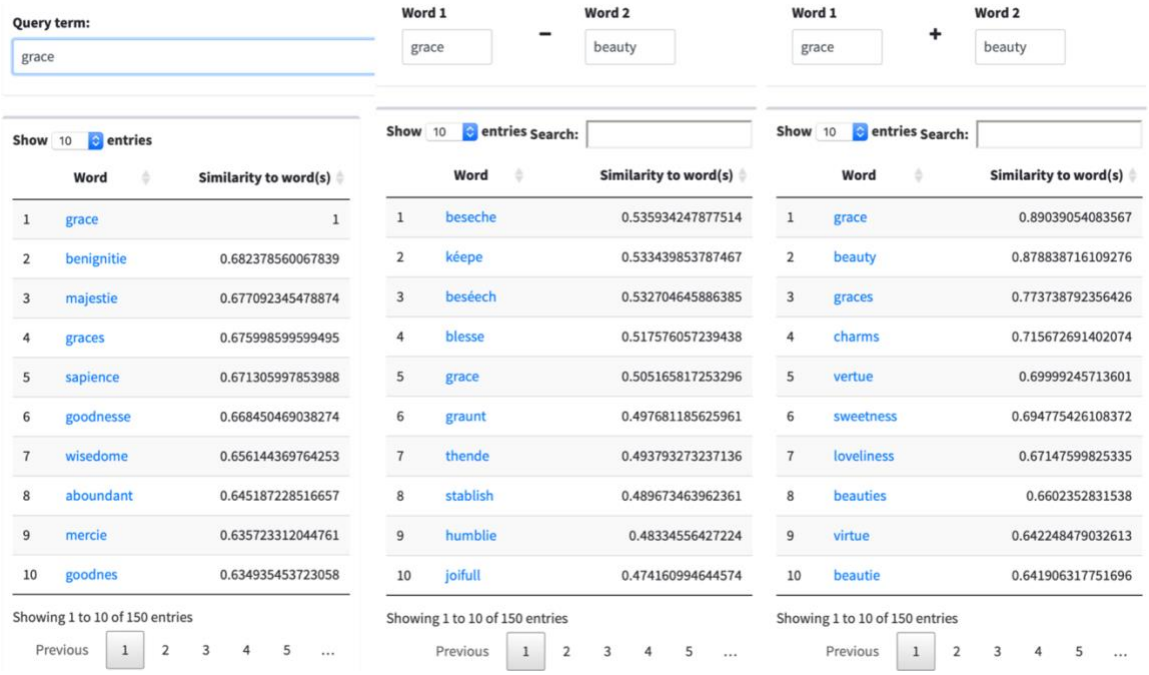
Using the Word Vector Interface, let us try these searches and consider the results.

Starting on the WWP home page, query "queen" and "king" using the "WWO Full Corpus" model and compare results.

Use the "Operations" function to query "queen" + "king" in the "WWO Full Corpus" and compare to the previous results.

What happens when you search "king" and "queen" plus one of their top ten closest words?

Now, select a cluster. Begin using the different search functions to learn more about the relationship between words in this cluster.
For example, Basic query: "beauty"
Operations (Addition): "beauty" + "virtue"
Operations (Subtraction): "beauty" - "pure" or "beauty" - "virtue"
Which results are surprising? Which match with what you were expecting to find? Which seem especially interesting or difficult to explain?
If you want to find more interesting clusters, hit "Reset."



Query term: grace

Show 10 entries

	Word	Similarity to word(s)
1	grace	1
2	benignitie	0.682378560067839
3	majestie	0.677092345478874
4	graces	0.675998599599495
5	sapience	0.671305997853988
6	goodnesse	0.668450469038274
7	wisedome	0.656144369764253
8	aboundant	0.645187228516657
9	mercie	0.635723312044761
10	goodnes	0.634935453723058

Showing 1 to 10 of 150 entries

Previous 1 2 3 4 5 ...

Word 1: grace − Word 2: beauty

Show 10 entries Search:

	Word	Similarity to word(s)
1	beseche	0.535934247877514
2	kéepe	0.533439853787467
3	beséech	0.532704645886385
4	blesse	0.517576057239438
5	grace	0.505165817253296
6	graunt	0.497681185625961
7	thende	0.493793273237136
8	stablish	0.489673463962361
9	humblie	0.48334556427224
10	joifull	0.474160994644574

Showing 1 to 10 of 150 entries

Previous 1 2 3 4 5 ...

Word 1: grace + Word 2: beauty

Show 10 entries Search:

	Word	Similarity to word(s)
1	grace	0.89039054083567
2	beauty	0.878838716109276
3	graces	0.773738792356426
4	charms	0.715672691402074
5	vertue	0.69999245713601
6	sweetness	0.694775426108372
7	loveliness	0.67147599825335
8	beauties	0.6602352831538
9	virtue	0.642248479032613
10	beautie	0.641906317751696

Showing 1 to 10 of 150 entries

Previous 1 2 3 4 5 ...

To generate a clustering list:

- the toolkit runs a clustering algorithm that randomly chooses 150 locations within the vector space (like throwing darts at the map)
- then, it makes a series of adjustments to those locations to move them closer to actual population centers, places where words are close to one another within the vector space
- here, you can get word clouds as visualizations

Validation

When we use our model, we get some results: predictable, provocative, and bizarre.
We need to test it to see whether that is a useful representation. To validate a model, we can ask:

- Are your results consistent across models and corpora?
- Did you try using the same parameters? Different parameters? Observe variations within the same corpus. Training a model is a probabilistic process: we will not get identical results from model to model, even on exactly the same corpus
- Do you get likely word groupings? If the answer is no, the model might not be very useful. Maybe the corpus is too small (less than one million words), or it was trained incorrectly
- When you work with vector math, do results change? Try all these: addition, subtraction, analogies
- Do analogies for common, not research words work likely?
- Try negative sampling is a way to reduce that work.

Discussion and Questions

Let us keep in mind that we are working with plain text, which does not allow for the analysis of semantic constructs. What we observe about word vectors, what they represent, and how they work is not universal, but local and situational. While we can learn a lot from these experiments, we can also go back and revisit the collection, parameters, and number of iterations.

- What questions could we also ask, based on the examples we had?
- What kinds of explanation do you think are ideal for different audiences: students; readers; project collaborators?

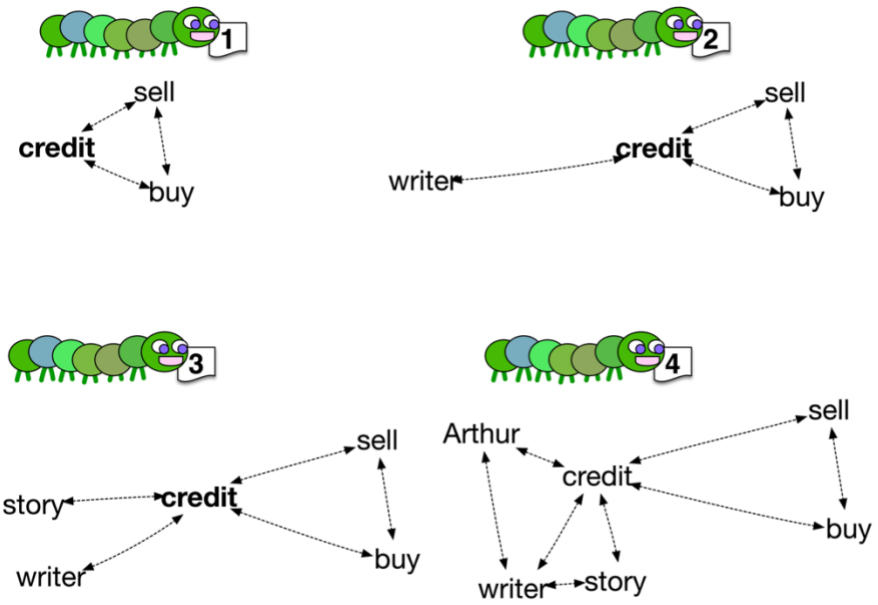
Keep a lab notebook: what you tried, what worked, what did not work. When you go write an article or do a presentation, you have documentation, which is like footnotes. You can also take screenshots of the various steps in your research.

How to Prepare Your Corpus

The size of the corpus should be greater than one million words.
To build a corpus, you can see ArchiveGrid
https://www.oclc.org/research/themes/research-collections/archivegrid.html

Testimonials

Sarah Connell, Northeastern University, studies the language of credit. One problem was the regularization of early modern language. For a word like credit, text analysis via word vectors might observe some instances where that word is associated with words like "sell" and "buy," and so it moves credit closer to those words: it adds information that makes an association between these words. In other cases, it observes some other instances where "credit" is associated with words like "story," "writer," and Arthur. All connections are detected, and their strength is also perceived in the system.



This is a visualization of Sarah Connell's iterations and negative sampling experiments

Adding and subtracting terms using word embedding models
(Woman-Man) (we're subtracting from it)+King (adding a third term)
Noun-noun=0Adult-adult=0Human-human=0 monarch
Female-male=Female-male (it is a vector about femaleness)

(affluence-poverty)+hockey=Lacrosse
It is a vector about richness, trained on Google News

Bonus question: what happens if we try this query?
Virtue - riches + learning =?

To sum up

More iterations are better; one iteration lasts one hour long, approximately. When you are training a model, the computer cannot do anything else. Good scholarship relies on forty iterations approximately.

Learn More About Word Vectors and the Women Writers Project

Tools for working with word embedding models are the following:

- word embedding algorithms
    - Word2Vec, developed by Tomas Mikolov at Google
      https://github.com/tmikolov/word2vec
    - GLoVE, developed by a research group at Stanford
      https://github.com/stanfordnlp/GloVe

- to run those algorithms on our data, we need:
    - the WordVectors package, written in R by Ben Schmidt
      https://github.com/bmschmidt/wordVectors
    - the GenSim package (written in Python by a Czech researcher, Radim Řehůřek
      https://radimrehurek.com/gensim
- to run these programs on our computer, we need an environment where the programming language (R, Python) can operate:
    - RStudio
    - Jupyter Notebooks

For general, quick word frequency analysis and concordances, see Voyant Tools
https://voyant-tools.org

Feedback form

Please complete this post-workshop survey. Your responses will help us to evaluate the effectiveness of the formats and themes http://bit.ly/s20_dh_feedback. Thank you!

Contact Information

You can contact me to follow up with questions:

Caterina Agostini caterina.agostini@rutgers.edu