Relative Semantic Analysis

Assessing political speeches can be fun! But how do you do it? One way to do it is to look at a candidate's words and see what they mean *relative* to other things that they say. Relative semantic analysis (RSA) is a computational tool that does just that. Please see below for more deals on RSA.

What is RSA?

Relative semantic analysis (RSA) is an approach to assessing the similarity between two n-grams within a thematically related corpus. RSA functions by considering the frequencies of any given n-gram's co-occurring words *relative* to another n-gram's co-occurring words. RSA produces a final output that is the relative semantic similarity between the two considered n-grams. This value ranges from 0 (no similarity) to 1 (semantically, the same).

What is a thematically related corpus?

A thematically related corpus is any collection of texts (documents) that the researcher considers to be identifiable under a common theme. Thus, "Hillary Clinton's responses to questions during the debates prior to the first caucus" would be considered a *thematically related corpus*. Similarly, "the Republican candidates' responses to questions during the debates prior to the first caucus" would also be considered *a thematically related corpus*. And also "10^{th} grade high-school science text-books' opening paragraphs" would also be considered *a thematically related corpus*.

What is a n-gram?

A n-gram is a string of adjacent words occurring within the thematically related corpus. For instance, given the sentence "a n-gram is a string of adjacent words", we can produce the following 2-word n-grams (or, bi-grams):
a n-gram,
bigram is,
is a,
a string,
string of,
of adjacent,
adjacent words.

A *tri-gram* is similar to a bi-gram but consists of three adjacent words. A *quad-gram* consists of four adjacent words. When words are considered individually, they are referred to as a *uni-gram*.

How does RSA work?

Let's consider the following example. We wish to compare the relative similarity of the following two uni-grams:

help and *governor*.

Further, we wish to evaluate this similarity relative to the theme of

Barak Obama's responses to questions during the debates prior to the first caucus.

In other words, how are the words *help* and *governor* related, relative to their use in *Barak Obama's responses to questions during the debates prior to the first caucus*.

RSA's procedure begins by calling the first document in the thematically related corpus. If the target word (i.e. *help*, in this example) does *not* occur in the document then the second document is called. This procedure continues until a document is found where the target word occurs. When a document is found with the target word, all the *other* words in that document (i.e., all the words except for *help*) are stored. Following this action, the next document in the corpus is called and the procedure continues until the corpus is exhausted.

The above procedure produces all the words in the corpus that *co-occur* with the target word. Many words will appear more than once. The RSA procedure accounts for this multiple appearance by calculating the frequency (i.e. how often it occurs)of each uniquely occurring word (or *type*). These types and their relative frequencies are then stored.

The entire above procedure is then rerun for the second target word (i.e., *governor*). Of course, *governor* is likely to have some co-occurring words that the word *help* did not. For these instances, a frequency of 0 is added. Thus, if the word *save* occurs 20 times for the word *help* occurs 0 times for the word *governor* then the word *help* is stored as |save (20, 0)|. When all the co-occurring words in the corpus are considered the frequencies for a line of word/frequency values, forms a *vector*. The vector length is equal to the number of types that co-occur with either target words (e.g., *help* and *governor*). To find the similarity between the two vectors, the *co-sine* is calculated.

cos = |A . B| / square root||A|| ||B|| (where A and B are the vectors representing the words *help* and *governor*). In this case, the cosine will be .216, indicating a relatively low similarity.

How does RSA-Nearest Neighbor work?

RSA-Nearest Neighbor works in much the same way as RSA. The major difference is that RSA-Nearest Neighbor considers only one n-gram at a time. The vector created by the n-gram during RSA processing is ordered by frequency. The word with the highest frequency is considered the *nearest neighbor*. As long as a word occurs in the corpus it will have a nearest neighbor; however, being the nearest neighbor does not *guarantee* that the target word and the nearest neighbor are similar in meaning. The nearest neighbor is

simply the most commonly co-occurring word, but that co-occurrence may be highly infrequent.

To take an example, the nearest neighbor for the word *life* relative to *the McCain corpus* is the word *war*. However, the similarity between *war* and *life* for the same corpus is a relatively modest .388. Meanwhile, the word *life* relative to *Clinton's corpus* produces a nearest neighbor of *well*. For these two words (i.e., *life* and *well*) the similarity is high, .723.

Remember that a perfect match is 1.0 and no match at all is 0.0.

How does RSA differ from similar tools such as Latent Semantic Analysis (LSA)?

Both RSA and LSA produce values of similarity between two or more words. However, operationally, the two approaches differ in a number of important ways.

LSA typically requires a large corpus of a few thousand documents. RSA is designed to operate with a corpus of a few hundred documents. In this project, we are dealing with relatively small corpora (note: one *corpus* but two *corpora*). As we simply don't have 1000s of documents about the recent political debates (because there were not 1000s of debates, although it may have felt like there were) then we can't make a super huge corpus that LSA needs. For that reason, we developed RSA which works very nicely on small corpora.

LSA documents are typically 100-500 words in length. RSA documents need only be two words long and typically they are about 100 words long. This is useful because debate responses are often very short indeed (although, again, they sometimes seem like they go on forever!)

LSA requires a multi-dimensional space that represents a type (in LSA speak the word used is *term*) by document matrix. In this matrix, the frequencies of all terms as they occur in their documents are recorded. Where no frequencies occur, a value of 0 is recorded for the corresponding cell. The subsequent vector created for any given word is as long as the number of documents in the corpus and typically features a majority of 0s. To reduce the dimensionality and the large 0-count of LSA, a mathematical procedure called singular value decomposition (SVD) is initiated. SVD effectively squeezes the cells in the vector together. Thus, a vector of a few thousand cells with a high frequency of 0s becomes a vector of a few hundred cells with a low frequency of 0s. LSA then compares vectors in the same way as RSA.

Unlike LSA, RSA does not create a term by document space, nor does it incorporate SVD. Because RSA functions with just a few hundred documents, there is no need to reduce the vector length. Also, because RSA (by definition) cannot have a frequency of 0 for both of the n-grams being compared there is no need to conduct SVD. That is, any values of 0 are meaningful in RSA and need to be retained.

One of the great benefits of any kind of semantic analysis (RSA or LSA) is that words can be compared across a range of documents without those words having to co-occur themselves. That is, the words *cat* and *dog* may never occur in the same document; however, as both words occur in documents that have the word *pet*, we can assess the similarity between *cat* and *dog* (because they both co-occur with the word *pet*). LSA approximates this similarity between non-co-occurring words (i.e., what is called *second order co-occurrences*) through the SVD procedure. Second order co-occurrence for RSA results from the raw frequency count, which is conducted independently of the documents. That is, in RSA, documents dictate whether types will be considered; however, the frequency count is for all types independent of which document they stemmed from.

For instance, consider a corpus of two documents: document A consists of two words *dog* and *pet*; document B also consists of two words *cat* and *pet*. Assessing the similarity between *dog* and *cat* in this corpus results in the following vectors

Dog: C [pet (1)]
Cat: C [pet (1)]

Dog/cat = pet (1,1)

= cosine of 1.00

Thus, while neither *cat* nor *dog* appear together, their co-occurrence with *pet* results in a high similarity value.

One major *dis*advantage of LSA is that it does not consider syntax or negation. For LSA, *well* and *not well* record a very high similarity (indeed generally a value of 1.0). Similarly, the sequence *can we* and *we can* would generate a value of 1.0 in LSA. In RSA, *well/not well* generates a similarity of just .28 for the *All corpus*. And for *can we/we can*, the similarity is .54 for the *All corpus* (Obama's corpus generates a similarity of .458 for *can we/we can*, suggesting the difference is greater from Obama's perspective than from the candidates as a whole). RSA is able to distinguish these similarities because it considers bi-grams as complete units. Thus, for RSA, *we can* is *we-can* and *can we* is *can-we*. Subsequently, the co-occurring words for each bi-gram will be different, and the value of similarity generated is less likely to be 1.0.

High frequency words

In semantic analysis (both RSA and LSA), high frequency words such as *the* and *to* can be problematic. LSA typically solves the problem of high frequency words by simply removing them at source. RSA is slightly different, removing high frequency words upon procedural completion. The contrast is important because the RSA approach leaves much of the benefit of high frequency words while losing much of the problem.

In the current version of the WWC 1.0 tool, a *stop list* is created of all the most commonly occurring words across the *ALL-corpus*. Any words with a probability value of > .01 are tagged. While this value may seem high, it amounts to only 159 words. RSA

operates oblivious to the stop list; however, *before* nearest neighbors are displayed on the tool's inter-face and before vectors are compared, all types and their corresponding frequencies that occur on the stop list are removed. As such, LSA and RSA may appear affectively the same in this respect. However, the real benefit of the stop list occurring last in the procedure is relative to *non* uni-gram comparisons. Consider a comparison of *you have to* and *we have to*. While each of these individual words (i.e., *we* and *have* and *to*) occurs on the stop list, RSA does *not* have on the stop list the instance of *you-have-to* or *we-have-to*. As such, the subtle difference between these n-grams (including their high frequency words) can be considered. Consequently, we can see that *you-have-to*/*we-have-to* has a similarity value of .306 relative to Obama, but .531 relative to Clinton: a result suggesting that Clinton views these n-grams as more similar than Obama.