Tool for the Automatic Analysis of Cohesion (TAACO)

Quick Start Guide and User Manual for TAACO 2.0.4 (updated 5-9-2018)

Kristopher Kyle, University of Hawaii at Manoa
Scott Crossley, Georgia State University

This document is intended to assist users of TAACO 2.0.4. It includes a brief explanation of how to get started with the tool, a description of how indices within the tool are calculated, and a section outlining "How to do a study with TAACO". If this is the first time using TAACO, it is advisable to read this entire document thoroughly, paying particular attention to the "How to Conduct a Study with TAACO" section. Additional information about TAACO is included in the supplementary Index Description Spreadsheet (available at www.kristopherkyle.com).

Please cite the following article when using TAALES in your work:

Crossley, S. A., Kyle, K., & McNamara, D. S. (2016). The tool for the automatic analysis of text cohesion (TAACO): Automatic assessment of local, global, and text cohesion. *Behavior Research Methods 48*(4), pp. 1227-1237. doi:10.3758/s13428-015-0651-7

What is new in TAACO 2.0.4?

TAACO 2.0.4 represents a substantive update to previous versions of TAACO. TAACO 2.0.4 introduces the following new/updated features:

- LSA, LDA, and word2vec semantic overlap features for sentences and paragraphs
- LSA, LDA and word2vec semantic overlap between target texts and source texts
- Keyword overlap between target texts and source texts for unigrams, bigrams, and trigrams
- Improved word disambiguation (e.g., *so* as an adverb versus *so* as a connective)

These updates provide more accurate and more intuitive calculations of cohesion. We suggest all TAACO users upgrade to version 2.0.4.

Compatibility

TAACO 2.0.4 works on Mac OSX, 64-bit versions of Windows 7, 8, and 10, and 64-bit Linux Ubuntu. Compiled versions of TAACO are not supported on 32-bit versions of Windows or Linux Ubuntu[1].

Getting Started

First, download the version of TAALES that is appropriate for your operating system. Second, download and install the version of the Java Development Kit (JDK) (JDK 8 or higher) that is

[1] Advanced users with some programming knowledge should be able to use the Python 2.7 version of TAACO on 32-bit systems, but technical support is not offered.

appropriate for your operating system. Note that this is NOT the same version of Java that is installed by default on most operating systems.

Input Text Formatting

TAACO is designed to process plain text (.txt) files with UTF-8 encoding. This is a very common encoding type, though not all word processing software will default to UTF-8 encoding. A common reason for TAACO to malfunction is text encoding. If problems occur during the use of TAACO (and particularly if an error message that includes ‘ascii codec can’t encode characters in position…’), then it is important to ensure that a) filenames only include “normal” English characters and b) texts are in UTF-8 format.

Options

Index Options

TAACO 2.0.4 allows users to calculate all possible indices or choose to only calculate particular index types. Indices included in TAACO 2.0.4 are described in the “Indices” section below. Further details can be found in the TAACO 2.0.4 Index Description Spreadsheet, which is freely available at www.kristopherkyle.com.

Diagnostic Output Options

TAACO 2.0.4 provides two types of diagnostic output designed to help users a) determine whether their data was processed correctly, b) understand how particular indices were calculated, and c) find examples of calculated linguistic features. The first (“diagnostic file”) includes basic information for each text such as the number of sentences, paragraphs, running word lemmas (tokens), and unique word lemmas (types) found in each input text. The second (“tagged files”) contains sentence breaks, paragraph breaks, and a variety of lemma and part of speech information for each word in the text.

If either of the diagnostic output options are selected, a folder is created that includes the user-selected output filename (e.g., “results”) with “_diagnostic” appended to it (e.g., “results_diagnostic”. All diagnostic files are written to this folder.

TAACO 2.0.4 Indices

TAACO reports 194 indices in seven main categories: TTR and Density, Lexical Overlap (sentences), Lexical Overlap (paragraphs), Semantic Overlap, Connectives, Givenness, and Source Text Similarity. The calculation of these indices is discussed below. A description of each index can be found in the Index Description Spreadsheet.

Note: All indices in TAACO 2.0.4 are calculated using the lemma form of each word (e.g., the lemma form of *is*, *am*, *are*, *was*, *were*, and *be* is *be*).

TTR and Density

TAACO 2.0.4 calculates 15 indices related to TTR (13 indices) and Density (2 indices). These are briefly described below.

TTR. Type-token ratio (TTR) indices provide a measure of lexical diversity. TTR is calculated by dividing the number of unique lemmas in a text and dividing it by the number of total lemmas in a text. TTR indices are available for single lemmas, for bi-grams (two word phrases), tri-grams (three word phrases). See Table 1 for an example of how TTR scores are calculated. TAACO reports variants of the TTR index for various part of speech categories (e.g., nouns and verbs). TAACO also report moving average TTR (MATTR), which comprises the average TTR score for overlapping 50-word sequences in a text (e.g., words 1-50, 2-51, 3-52, etc.; see Covington & McFall, 2010).

Table 1
TTR example

Input text	This is a cool sentence. This sentence is even cooler. Those were both really cool sentences.
Lemmatized text	This be a cool sentence. This sentence be even cooler. Those were both really cool sentence.
Number of unique words (types)	10
Number of total words (tokens)	16
TTR score	**10/16 = .625**

Lexical Density. Lexical density indices indicate the proportion of the text that consists of content words (nouns, lexical verbs, adjectives, and adverbs derived from adjectives). These are calculated by dividing the number of content words in a text by the number of total words in a text. Indices are available for both tokens and types.

Sentence Overlap

TAACO 2.0.4 calculates six basic types of sentence overlap indices. Each of these is described below. Table 2 includes a sample text at three stages of processing. First, a raw text is imported into TAACO. In the second stage, the text is lemmatized and separated into sentences. In the third stage, each sentence is refined so that it includes only unique lemmas (types). Overlap indices use the version of the text included in Step 3. Each description of the overlap indices will refer to the sample text in Table 2.

Table 2
Sentence overlap examples

Step 1: Raw input text	Automatic text analysis can be very convenient. It is, however, important for users to understand how each index is calculated. In many cases, there are multiple ways that each index could be calculated. It is also important to understand what constructs an index is purported to measure.	
Step 2: Lemmatized text (separated into sentences)	LS 1	automatic text analysis can be very convenient
	LS 2	it be however important for user to understand how each index be calculate
	LS 3	in many case there be multiple way that each index could be calculate
	LS 4	it be also important to understand what construct a index be purported to measure
Step 3: Unique lemmas (tokens) in each sentence	LTS 1	automatic text analysis can be very convenient
	LTS 2	it be however important for user to understand how each index calculate
	LTS 3	in many case there be multiple way that each index could calculate
	LTS 4	it be also important to understand what construct a index purported measure

Adjacent sentence overlap. Adjacent sentence overlap calculates the average amount of words that are repeated between sentences. In each sentence that is followed by another sentence (i.e., LTS 1, LTS 2, and LTS 3) each word is checked in order to determine whether it occurs in the following sentence. For each word that occurs in the following sentence at least once, the overlap count increases by one. In LTS 1, for example, only the word *be* occurs in the following sentence (LTS 2). The total index score is computed as the total number of words with adjacent sentence overlap divided by the number of words considered (i.e., the sum of the number lemma types in LTS 1, LTS 2, and LTS 3). Table 3 includes the calculation of adjacent sentence overlap for the example text.

Table 3
Adjacent sentence overlap calculation

Overlap check	Overlapping lemmas	Overlap Count	Number of lemma types
LTS 1 -> LTS 2	*be*	1	7
LTS 2 -> LTS 3	*be*, *calculate*, *each*, *index*	4	12
LTS 3 -> LTS 4	*be*, *index*	2	12
	Sum	7	31
	Adjacent sentence overlap score	7 / 31 = 0.2258	

Adjacent sentence overlap (sentence normed). Adjacent sentence overlap (sentence normed) is calculated in the same manner as adjacent sentence overlap, except that the denominator is the number of sentences considered. In short, the total index score is computed as the total number of words with adjacent sentence overlap divided by the number of sentences considered. Table 4 includes the calculation of adjacent sentence overlap (sentence normed) for the example text.

Table 4
Adjacent sentence overlap (sentence normed) calculation

Overlap check	Overlapping lemmas	Overlap Count	Number of sentences considered
LTS 1 -> LTS 2	*be*	1	1
LTS 2 -> LTS 3	*be*, *calculate*, *each*, *index*	4	1
LTS 3 -> LTS 4	*be*, *index*	2	1
	Sum	7	3
	Adjacent sentence overlap score	7 / 3 = 2.333	

Binary adjacent sentence overlap. Binary adjacent sentence overlap calculates how many adjacent sentences include any overlapping items. The total index score is computed as the total number of sentences that include words that occur in the following sentence divided by the number of sentences considered. Table 5 includes the calculation of binary adjacent sentence overlap for the example text.

Table 5
Binary adjacent sentence overlap calculation

Overlap check	Overlapping lemmas	Overlap Count	Number of sentences considered
LTS 1 -> LTS 2	*be*	1	1
LTS 2 -> LTS 3	*be*, *calculate*, *each*, *index*	1	1
LTS 3 -> LTS 4	*be*, *index*	1	1
	Sum	3	3
	Adjacent sentence overlap score	3 / 3 = 1	

Adjacent two-sentence overlap. Adjacent two-sentence overlap calculates the average amount of words that are repeated between sentences. In each sentence that is followed by at least two sentences (i.e., LTS 1 and LTS 2) each word is checked in order to determine whether it occurs in the following two sentences. For each word that occurs in the two following sentences at least once, the overlap count increases by one. In LTS 1, for example, only the word *be* occurs in the two following sentences (LTS 2). The total index score is computed as the total number of words with adjacent two-sentence overlap divided by the number of words considered (i.e., the sum of the number lemma types in LTS 1, LTS 2). Table 6 includes the calculation of adjacent two-sentence overlap for the example text.

Table 6
Adjacent two-sentence overlap calculation

Overlap check	Overlapping lemmas	Overlap Count	Number of lemma types
LTS 1 -> LTS 2 and LTS 3	*be*	1	7
LTS 2 -> LTS 3 and LTS 4	*it, be*, *important*, *to*, *understand*, *each*, *index*, calculate	8	12
Sum		9	19
Adjacent sentence overlap score		9 / 19 = .4737	

Adjacent two-sentence overlap (sentence normed). Adjacent two-sentence overlap (sentence normed) is calculated in the same manner as adjacent two-sentence overlap, except that the denominator is the number of sentences considered. In short, the total index score is computed as the total number of words with adjacent two-sentence overlap divided by the number of sentences considered. Table 7 includes the calculation of adjacent two-sentence overlap (sentence normed) for the example text.

Table 7
Adjacent two-sentence overlap (sentence normed) calculation

Overlap check	Overlapping lemmas	Overlap Count	Number of sentences considered
LTS 1 -> LTS 2 and LTS 3	*be*	1	1
LTS 2 -> LTS 3 and LTS 4	*it, be*, *important*, *to*, *understand*, *each*, *index*, calculate	8	1
Sum		9	2
Adjacent sentence overlap score		9 / 2 = 4.5	

Binary two-sentence overlap. Binary adjacent two-sentence overlap calculates how many adjacent sentences include any overlapping items. The total index score is computed as the total number of sentences that include words that occur in the following two sentences divided by the number of sentences considered. Table 8 includes the calculation of binary adjacent two-sentence overlap for the example text.

Table 8
Binary two-sentence overlap calculation

Overlap check	Overlapping lemmas	Binary overlap Count	Number of sentences considered
LTS 1 -> LTS 2 and LTS 3	*be*	1	1
LTS 2 -> LTS 3 and LTS 4	*it*, *be*, *important*, *to*, *understand*, *each*, *index*, calculate	1	1
Sum		2	2
Adjacent sentence overlap score		2 / 2 = 1	

Paragraph Overlap
Paragraph overlap indices are identical to sentence overlap indices, except that adjacent paragraphs are examined instead of sentences. Table 9 includes a sample text at three stages of processing. First, a raw text is imported into TAACO. In the second stage, the text is lemmatized and separated into paragraphs. In the third stage, each paragraph is refined so that it includes only unique lemmas (types). Paragraph overlap indices use the version of the text included in Step 3. Each description of the overlap indices will refer to the sample text in Table 9.

Table 9
Paragraph overlap examples

Step 1: Raw input text		Automatic text analysis can be very convenient. It is, however, important for users to understand how each index is calculated. In many cases, there are multiple ways that each index could be calculated. It is also important to understand what constructs an index is purported to measure. Conducting an analysis without understanding how indices are measured will at the very least make it difficult to interpret the results. Worse, this can also lead a researcher to make unwarranted conclusions about the data. This user manual has been created to make TAACO indices easier to understand. It is hoped that this will help both novices and experts conduct high impact studies.
Step 2: Lemmatized text (separated into paragraphs)	LP 1	automatic text analysis can be very convenient it be however important for user to understand how each index be calculate
	LP 2	in many case there be multiple way that each index could be calculate it be also important to understand what construct a index be purported to measure
	LP 3	conduct a analysis without understand how index be measure will at the very least make it difficult to interpret the result worse this can also lead a researcher to make unwarranted conclusion about the datum

	LP 4	this user manual have be create to make taaco index easier to understand it be hope that this will help both novice and expert conduct high impact study
Step 3: Unique lemmas (tokens) in each paragraph	LTP 1	automatic text analysis can be very convenient it however important for user to understand how each index calculate
	LTP 2	in many case there be multiple way that each index could calculate it also important to understand what construct a purported measure
	LTP 3	conduct a analysis without understand how index be measure will at the very least make it difficult to interpret result worse this can also lead researcher unwarranted conclusion about datum
	LTP 4	this user manual have be create to make taaco index easier understand it hope that this will help both novice and expert conduct high impact study

Adjacent paragraph overlap. Adjacent paragraph overlap calculates the average amount of words that are repeated between paragraphs. In each paragraph that is followed by another paragraph (i.e., LTP 1, LTP 2, and LTP 3) each word is checked in order to determine whether it occurs in the following paragraph. For each word that occurs in the following paragraph at least once, the overlap count increases by one. In LTP 1, for example, eight words (*be*, *it*, *important*, etc.) occur in the following paragraph (LTP 2). The total index score is computed as the total number of words with adjacent paragraph overlap divided by the number of words considered (i.e., the sum of the number lemma types in LTP 1, LTP 2, and LTP 3). Table 10 includes the calculation of adjacent paragraph overlap for the example text.

Table 10
Adjacent paragraph overlap calculation

Overlap check	Overlapping lemmas	Overlap Count	Number of lemma types
LTP 1 -> LTP 2	*be, it, important, to, understand, each, index, calculate*	8	18
LTP 2 -> LTP 3	*be, index, it, also, to, understand, a, measure*	8	22
LTP 3 -> LTP 4	*conduct, understand, index, be, will, make, it, to, this*	9	30
	Sum	25	70
	Adjacent sentence overlap score	25 / 70 = .3571	

Adjacent paragraph overlap (paragraph normed). Adjacent paragraph overlap (paragraph normed) is calculated in the same manner as adjacent paragraph overlap, except that the denominator is the number of paragraphs considered. In short, the total index score is computed as the total number of words with adjacent paragraph overlap divided by the number of paragraphs considered. Table 11 includes the calculation of adjacent paragraph overlap (paragraph normed) for the example text.

Table 11
Adjacent paragraph overlap (paragraph normed) calculation

Overlap check	Overlapping lemmas	Overlap Count	Number of paragraphs considered
LTP 1 -> LTP 2	*be, it, important, to, understand, each, index, calculate*	8	1
LTP 2 -> LTP 3	*be, index, it, also, to, understand, a, measure*	8	1
LTP 3 -> LTP 4	*conduct, understand, index, be, will, make, it, to, this*	9	1
Sum		25	3
Adjacent sentence overlap score		25 / 3 = 8.3333	

Binary adjacent paragraph overlap. Binary adjacent paragraph overlap calculates how many adjacent paragraphs include any overlapping items. The total index score is computed as the total number of paragraphs that include words that occur in the following paragraph divided by the number of paragraphs considered. Table 12 includes the calculation of binary adjacent paragraph overlap for the example text.

Table 12
Adjacent sentence overlap calculation

Overlap check	Overlapping lemmas	Overlap Count	Number of paragraphs considered
LTP 1 -> LTP 2	*be, it, important, to, understand, each, index, calculate*	1	1
LTP 2 -> LTP 3	*be, index, it, also, to, understand, a, measure*	1	1
LTP 3 -> LTP 4	*conduct, understand, index, be, will, make, it, to, this*	1	1
Sum		3	3
Adjacent sentence overlap score		3 / 3 = 1	

Adjacent two-paragraph overlap. Adjacent two-paragraph overlap calculates the average amount of words that are repeated between paragraphs. In each paragraph that is followed by at least two paragraphs (i.e., LTP 1 and LTP 2) each word is checked in order to determine whether it occurs in the following two paragraphs. For each word that occurs in the two following paragraphs at least once, the overlap count increases by one. In LTP 1, for example, twelve words occur, in the two following paragraphs (LTP 2). The total index score is computed as the total number of words with adjacent two-paragraph overlap divided by the number of words considered (i.e., the sum of the number lemma types in LTP 1, LTP 2). Table 13 includes the calculation of adjacent two-paragraph overlap for the example text.

Table 13
Adjacent two-paragraph overlap calculation

Overlap check	Overlapping lemmas	Overlap Count	Number of lemma types
LTP 1 -> LTP 2 and LTP 3	*analysis, can, be, very, it, important, to, understand, how, each, index, calculate*	12	18
LTP 2 -> LTP 3 and LTP 4	*be, that, index, it, also, to, understand, a, measure*	9	22
	Sum	21	40
	Adjacent sentence overlap score	21 / 40 = .5250	

Adjacent two-paragraph overlap (paragraph normed). Adjacent two-paragraph overlap (paragraph normed) is calculated in the same manner as adjacent two-paragraph overlap, except that the denominator is the number of paragraphs considered. In short, the total index score is computed as the total number of words with adjacent two-paragraph overlap divided by the number of paragraphs considered. Table 14 includes the calculation of adjacent two-paragraph overlap (paragraph normed) for the example text.

Table 14
Adjacent two-paragraph overlap (paragraph normed) calculation

Overlap check	Overlapping lemmas	Overlap Count	Number of paragraphs considered
LTP 1 -> LTP 2 and LTP 3	*analysis, can, be, very, it, important, to, understand, how, each, index, calculate*	12	1
LTP 2 -> LTP 3 and LTP 4	*be, that, index, it, also, to, understand, a, measure*	9	1
	Sum	21	2
	Adjacent sentence overlap score	21 / 2 = 10.5	

Binary two-paragraph overlap. Binary adjacent two-paragraph overlap calculates how many adjacent paragraphs include any overlapping items. The total index score is computed as the total number of paragraphs that include words that occur in the following two paragraphs divided by the number of paragraphs considered. Table 15 includes the calculation of binary adjacent two-paragraph overlap for the example text.

Table 15
Binary two-paragraph overlap

Overlap check	Overlapping lemmas	Binary overlap Count	Number of paragraphs considered
LTP 1 -> LTP 2 and LTP 3	*analysis, can, be, very, it, important, to, understand, how, each, index, calculate*	1	1
LTP 2 -> LTP 3 and LTP 4	*be, that, index, it, also, to, understand, a, measure*	1	1
	Sum	2	2
Adjacent sentence overlap score		2 / 2 = 1	

Semantic Overlap

An important element of discourse cohesion in terms of NLP techniques is to what extent computational models of semantic memory (Cree & Armstrong, 2012) are capable of highlighting underlying semantic relations in texts. These computational models rely on unsupervised learning methods that measure cohesion between textual fragments (Bestgen & Vincze, 2012). Common models include semantic vector spaces using Latent Semantic Analysis (LSA; Landauer et al., 1998), topic distributions in Latent Dirichlet Allocation (LDA, Blei et al., 2003), and word2vec vector space representations (Mikolov et al., 2013).

Latent Semantic Analysis. LSA (Landauer et al., 1998) is a mathematical optimization for representing word meanings in an orthogonal vector space created through an unsupervised learning method applied on a large text corpus. The vector space model is based on word co-occurrences within documents which establish relationships between concepts. LSA builds a term-document matrix, which is normalized using log-entropy, followed by a singular-value decomposition (SVD; Golub & Reinsch, 1970) and a projection over the most representative *k* dimensions. LSA has been subjected to extensive psychological experiments which suggest it models linguistic and cognitive knowledge. The LSA model used in TAACO 2.0 was created using the Stochastic SVD from Apache Mahout (Owen, Anil, Dunning, & Friedman, 2011) and considered several optimizations including ignoring stop-words and non-dictionary words, reducing inflected word forms to their corresponding lemmas, erasing words with low frequencies (i.e., 5 occurences), elimination of paragraphs with fewer than 20 words, and projection over 300 dimensions as suggested by Landauer, McNamara, Dennis, and Kintsch (2007).

Latent Dirichlet Allocation (LDA). LDA (Blei et al., 2003) is a generative probabilistic process in which documents are perceived as mixtures of multiple topics. Documents and words alike are topics distributions drawn from Dirichlet distributions and topics (i.e., latent variable in the model) are perceived as a Dirichlet distribution over the input vocabulary (Kotz, Balakrishnan, & Johnson, 2004). Similar to LSA, related concepts have similar topic probabilities based on underlying co-occurrence patterns. Our HDP model was trained using Mallet (McCallum, 2002) over 300 (to ensure equitability with the other models).

Word2vec. Word2vec (Mikolov et al., 2013) relies on a neural network model to represent words and phrases in a vector-space model with a limited number of dimensions - *k*.

Each word's embedding is computed using the context around it within the training dataset; thus, words co-occurring in similar contexts are represented closer, while words with dissimilar contexts are represented farther apart in different regions of the vector space. Our word2vec model was trained using GenSim (Řehůřek & Sojka, 2010) with Skip-Gram negative sampling, a window of 5 words and 300 dimensions.

Semantic Similarity Calculations. For each of these models (LSA, LDA, word2vec), TAACO 2.0 calculates the average similarity between progressive adjacent segments (sentences or paragraphs) in a text. All semantic models in TAACO 2.0 were trained on the newspaper and magazine sections of the Corpus of Contemporary American English (COCA; Davies, 2008). The size of these two corpora is around 230 million words (~117 million words in the magazine corpus and ~113 million words in the newspaper corpus). Prior to developing the semantic spaces, all function words were removed as were all identified non-English words (i.e., misspellings), Lastly, all words in the corpora were lemmatized.

All semantic models relied on the bag-of-words assumption in which word order is disregarded. TAACO semantic similarity indices are calculated using two methods for exploring adjacent text segments. In the first, similarity scores are calculated between segments (e.g., sentences) one and two, two and three, three and four, etc., until all progressive segment pairs have been examined. In the second method, similarity scores are calculated between segment one and the combination of segments two and three, then segment two and the combination of segments three and four, etc., until all progressive segments have been compared.

Connectives

TAACO 2.0.4 calculates 25 connectives indices. For a particular connectives index list, the occurrence of each item in the list in the input text is counted. After checking the entire text for each list item, the sum of the list item is divided by the total number of words in the text. See Table 16 for an example. Each list of connectives is included below. In some cases, ambiguous words are disambiguated using the Stanford Neural Network Dependency Parser (Chen & Manning, 2014) these are bolded in the lists below.

Table 16
Connectives example using the basic connectives list

input text	Jack and Jill went up the hill to fetch a pail of water. They put the bucket in the well, but there was no water. Jack and Jill then decided to hike down the hill to a stream instead.
basic connectives	for, and, nor, but, or, yet, so
input text with basic connectives identified	Jack **and** Jill went up the hill to fetch a pail of water. They put the bucket in the well, **but** there was no water **so** Jack **and** Jill decided to hike down the hill to a stream instead.
number of basic connectives	4
number of total words	39
basic connectives index score	**4 / 39 = .1026**

Connective lists used in TAACO 2.0.4

Basic connectives. *for, and, nor, but, or, yet, so*

Conjunctions. *and, but*

Disjunctions. *or*

Lexical subordinators. Lexical subordinators are identified using the "mark" tag in the Stanford dependency representation. The following words are examples of words that often receive this tag: *after, although, as, because, before, if, once, since, that, though, till, unless, until, whenever, wherever, whereas, whereupon, while*

Coordinating conjuncts. *yet, so, nor, however, therefore*

Addition. *and, also, besides, further, furthermore, too, moreover, in addition, then, another, indeed, likewise*

Sentence linking. *nonetheless, therefore, although, furthermore, whereas, nevertheless, whatever, for, however, besides, henceforth, then, yet, if, while, so, but, until, because, alternatively, meanwhile, when, since, notwithstanding, whenever, moreover, as, consequently, after*

Order. *to begin with, next, first, firstly, second, secondly, finally, in conclusion, above all,* ***before***, ***after***, *then*

Reason and purpose. *therefore, that is why, for this reason, for that reason, hence, because,* ***so****,* ***since****, as, because of, on account of, so that, consequently*

All causal connectives. *although, arise, arises, arising, arose, because, cause, caused, causes, causing, condition, conditions, consequence, consequences, consequent, consequently, due to, enable, enabled, enables, enabling, even then, follow that, follow the, follow this, followed that, followed the, followed this, following that, follows the, follows this, hence, made, make, makes, making, nevertheless, nonetheless, only if, provided that, result, results,* ***since****, so, therefore, though, thus, unless, whenever*

Positive causal connectives. *arise, arises, arising, arose, because, cause, caused, causes, causing, condition, conditions, consequence, consequences, consequent, consequently, due, enable, enabled, enables, enabling, even, follow, followed, following, follows, hence, if, made, make, makes, making, only, provided, result, results,* ***since****,* ***so****, then, therefore, this, thus*

Opposition. *but, however, nevertheless, otherwise, on the other hand, on the contrary, yet, still, maybe, perhaps, instead, except for, in spite of, despite, nonetheless, apart from, unlike, whereas*

Determiners. *a, an, the, this, that, these, those*

Demonstratives. In addition to basic demonstrative counts, TAACO 2.0.4 also includes indices that are sensitive to whether a demonstrative is attended (i.e., directly precedes a noun phrase; *This noun phrase is attended*) or unattended (functions as a noun phrase; *This is unattended*). Following are the determiner list: *this, that, these, those*

All additive connectives. *after all, again, all in all, also, alternatively, and, anyhow, as a final point, as well, at least, besides, but, by contrast, by the way, contrasted with, correspondingly, except that, finally, first, for example, for instance, fortunately, further, furthermore, however, in actual fact, in addition, in contrast, in fact, in other words, in sum, incidentally, instead, it follows, moreover, next, notwithstanding that, on one hand, on the contrary, on the one hand, on the other hand, or, otherwise, rather, secondly, similarly, summarizing, summing up, that is, thereupon, to conclude, to return to, to sum up, to summarize, to take an example, to these ends, to this end, too, well at any rate, whereas, yet*

All logical connectives. *actually, admittedly, after all, all in all, also, alternatively, although, and conversely, anyhow, anyway, arise from, arise out of, arises from, arises out of, arising from, arising out of, arose from, arose out of, as a final point, as a result, as well, at least, at this point, because, besides, but, by contrast, cause, caused, causes, causing, conditional upon, consequence, consequences, consequently, contrasted with, correspondingly, despite the fact that, due to, enable, enabled, enables, except that, finally, follow that, follow the, follow this, followed that, followed the, followed this, following that, follows the, follows this,* ***for****, fortunately, further, furthermore, hence, however, if, in actual fact, in any case, in any event, in case, in conclusion, in contrast, in fact, in order that, in other words, in short, in sum, incidentally, instead, it followed that, it follows, it follows that, likewise, moreover, nevertheless,* *next, nonetheless, nor, notwithstanding that, on condition that, on one hand, on the condition that, on the contrary, on the one hand, on the other hand, once again, or, otherwise, provided that, purpose of which, pursuant to, rather, secondly, similarly,* ***since***, ***so***, *summarizing, summing up, that is, that is to say, then, therefore, thereupon, though, thus, to conclude, to return to, to sum up, to summarize, to take an example, to that end, to these ends, to this end, to those ends, unless, well at any rate, whereas, while*

Positive logical connectives. *actually, after all, all in all, also, anyway, arise from, arise out of, arises from, arises out of, arising from, arising out of, arose from, arose out of, as a final point, as a result, as well, at least, at this point, because, besides, cause, caused, causes, causing, conditional upon, consequence, consequences, consequently, correspondingly, due to, enable, enabled, enables, finally, follow that, follow the, follow this, followed that, followed the, followed this, following that, follows the, follows this,* ***for***, *fortunately, further, furthermore, hence, if, in actual fact, in any case, in any event, in case, in conclusion, in fact, in order that, in other words, in short, in sum, incidentally, instead, it followed that, it follows, it follows that, likewise, moreover, next, on condition that, on one hand, on the condition that, on the one hand, once again, provided that, purpose of which, pursuant to, secondly, similarly,* ***since***, ***so***, *summarizing, summing up, that is, that is to say, then, therefore, thereupon, thus, to conclude, to return to, to sum up, to summarize, to take an example, to that end, to these ends, to this end, to those ends, well at any rate, while*

Negative logical connectives. *admittedly, alternatively, although, and conversely, anyhow, but, by contrast, contrasted with, despite the fact that, except that, however, in contrast, nevertheless, nonetheless, nor, notwithstanding that, on the contrary, on the other hand, or else, otherwise, rather, though, unless, whereas, yet*

Temporal connectives. *a consequence of,* ***after***, *again, all this time,* ***as***, *at last, at once, at the same time, at this moment, at this point,* ***before***, *by this time, earlier, finally, first, follow that, following that, from now on, further, immediately, in the meantime, instantly, it followed that, it follows, it follows that, later, meanwhile, next, now that, on another occasion, once more, presently, previously, secondly, simultaneously, since, so far, soon, suddenly, the consequence of, the consequences of, the last time, the previous moment, then, this time, throughout, to that end, until, up till that time, up to now, when, whenever, while*

Positive intentional connectives. ***by***, *desire, desired, desires, desiring, goal, goals, in order, made, make, makes, making, purpose, purposes,* ***so***, *to that end, to these ends, to this end, to those ends, want, wanted, wanting, wants*

All positive connectives. *actually,* ***after***, *again, all in all, also, and, anyway, arise, arises, arising, arose, as, at last, at least, at once, because,* ***before***, *besides,* ***by***, *cause, caused, causes, causing, condition, conditional upon, conditions, consequence, consequences, consequent, consequently, correspondingly, desire, desired, desires, desiring, due, earlier, enable, enabled, enables, enabling, even, finally, first, follow, followed, following, follows, for, fortunately, from now on, further, furthermore, goal, goals, hence, if, immediately, in actual fact, in addition, in any case, in any event, in case, in conclusion, in fact, in order, in other words, in short, in sum, incidentally, instantly, instead, later, likewise, made, make, makes, making,* *meanwhile, moreover, next, on another occasion, on one hand, once more, only, provided, presently, previously, purpose, purposes, result, results, secondly, similarly, simultaneously, since, so, soon, suddenly, summarizing, summing up, then, therefore, thereupon, throughout, thus, too, want, wanted, wanting, wants, well at any rate, when, whenever, while*

All negative connectives. *admittedly, alternatively, although, and conversely, anyhow, but, by contrast, contrasted with, despite the fact that, except that, however, in contrast, nevertheless, nonetheless, nor, notwithstanding that, on the contrary, on the other hand, or, otherwise, rather, though, unless, until, whenever, whereas, yet*

All connectives. *actually, admittedly,* ***after****, again, all in all, all this time, also, alternatively, although, and, anyhow, anyway, arise, arises, arising, arose, as, at last, at least, at once, at the same time, at this moment, at this point, because,* ***before****, besides, but,* ***by****, cause, caused, causes, causing, condition, conditional upon, conditions, consequence, consequences, consequent, consequently, contrasted with, correspondingly, desire, desired, desires, desiring, despite the fact that, due to, enable, enabled, enables, enabling, except that, finally, first, follow that, follow the, follow this, followed that, followed the, followed this, following that, follows the, follows this, fortunately, from now on, further, furthermore, goal, goals, hence, however, if, immediately, in actual fact, in addition, in any case, in any event, in case, in conclusion, in contrast, in fact, in order, in other words, in short, in sum, in the end, in the meantime, incidentally, instead, it followed that, it follows, it follows that, likewise, made, make, makes, making, meanwhile, moreover, nevertheless, next, nonetheless, nor, notwithstanding that, now that, on another occasion, on one hand, on the contrary, on the one hand, on the other hand, once more, or, otherwise, presently, previously, provided that, purpose of which, pursuant to, rather, secondly, similarly, simultaneously, since,* ***so****, summarizing, summing up, that is, the last time, the previous moment, then, therefore, thereupon, this time, though, throughout, thus, to conclude, to return to, to sum up, to summarize, to take an example, to that end, to these ends, to this end, to those ends, too, unless, until, up till that time, up to now, well at any rate, whenever, whereas, while, yet*

Givenness Givenness indices approximate the proportion of given information to new information by examining pronoun density, pronoun to noun ratios, and repeated content lemmas and pronouns. TAACO 2.0.4 calculates four indices related to givenness. Each is described in Table 17.

Table 17
Givenness indices in TAACO 2.0.4

Index	Calculation
pronoun density	number of third person pronouns divided by the total number of words
pronoun to noun ratio	the number third person pronouns divided by the total number of nouns
repeated content lemmas	the number of content words that are repeated at least once divided by the total number of words in the text
repeated content lemmas and pronouns	the number of content words and third person pronouns that are repeated at least once divided by the total number of words in the text

Source Text Similarity

Source similarity indices provide an overall evaluation of the similarity between the words in a source text and a target text, but do not differentiate between words that are paraphrased, words that are directly copied, and words that are text prominent. Key word overlap indices measure the degree to which important words and *n*-grams (i.e., bi-grams, tri-grams and quad-grams of *n* consecutive words) from the source text are found in the target text. TAACO 2.0.4 identifies key words and *n*-grams by comparing the relative frequency of these items in the source text to the relative frequency of the same items in the news and magazine sections of COCA. Any word or *n*-gram that occurs at least twice in the source text and that occurs relatively more frequently in the source text than in the reference corpus is preliminarily selected as a key word or *n*-gram. After the preliminary list has been compiled, the top 10% of words or *n*-grams (i.e., those that occur with the highest frequency in the source text as compared to COCA) are selected as key words or *n*-grams. TAACO 2.0.4 then calculates the proportion of words or *n*-grams in each target text that are key words or *n*-grams in the source text. In total, TAACO 2.0.4 calculates 23 key word overlap indices, as described below.

TAACO 2.0.4 calculates key word overlap for all words, bi-grams, tri-grams, and quad-grams. Part of speech (POS) statistics considering sensitive key overlap indices at the word level for nouns, adjectives, and verbs are also computed. Also included are key overlap indices for words that are verbs, nouns, adjectives, or nouns. At the *n*-gram level, POS sensitive key word overlap indices are calculated that allow for variable slots within the *n*-gram. For these indices, only words with particular parts of speech (e.g., nouns) are realized as concrete items, while other words are counted as variable slots. For example, the bigram “generalized reciprocity”, which consists of an adjective (“generalized”) and a noun (“reciprocity”) would be represented as “X reciprocity” (wherein “X” represents a variable slot). Such indices are available with the following parts of speech included as concrete items: nouns, adjectives, verbs, verbs and nouns as well as adjectives and nouns.

Additionally, for each semantic similarity calculation method outlined in a preceding section (i.e., LSA, LDA, and word2vec), a single source similarity index is calculated between a source text and a target text (e.g., a writing prompt and an essay written on that writing prompt).

How to Conduct a Study with TAACO

This section comprises an overview of how to conduct a study using TAACO. This is not meant to be comprehensive, but it should provide researchers an introduction into text analysis using automated tools.

1. Determine Research Questions

First and foremost, an automatic analysis tool such as TAACO is only useful if it helps a researcher address specific research questions. Therefore, having at least one concrete research question is vital to conducting meaningful research using a text analysis tool.

2. Learn About Indices

Before processing data with a text processing tool, it is important to determine what indices the tool measures and whether the calculated indices meets the needs of the researcher. For TAACO 2.0.4, this information can be found in the TAACO 2.0.4 Indices section above and the TAACO 2.0.4 Index Description Spreadsheet, which is available at www.kristopherkyle.com.

3. Choose Indices

After learning about the indices, a researcher should determine which indices will be informative for investigating the research questions. Avoiding the use of indices that are not clearly related to one's research questions will help make interpreting results more straightforward.

4. Format Input Texts

Before running a text analysis program, it is important to ensure that the texts to be analyzed are in the appropriate format (see the Input Text Formatting section above) and are largely free from spelling and punctuation errors. The algorithms that detect sentence boundaries, for example, will not work correctly if there is no white space (e.g., a space or new line character) between sentence ending punctuation and the beginning of the new sentence. For example, TAACO will count "This is a sentence.This is a second sentence." as a single sentence, and will also count "sentence.This" as a single word.

5. Process Data

After completing the steps above, data should be processed using TAACO 2.0.4's interface.

6. Check Data Processing Results

TAACO 2.0.4 provides two types of diagnostic output to help users ensure that their data was processed accurately. See the "Options" section above for more details. Before beginning a statistical analysis, it is important to spot check the data to ensure that a) words, sentences and paragraphs are being counted accurately and b) the lemmatizer and part of speech tagger is producing accurate results. Both of these can be checked using the diagnostic output features. If major problems are discovered with regard to a) or b), the likely problem is the format of the input texts.

7. Conduct Preliminary Statistical Tests to Meet Assumptions

With any statistical analysis, it is important for researchers to ensure that their data meets the assumptions of the statistical test(s) that are chosen. In many cases, analyses with TAACO indices involve statistical tests such as multiple regression, discriminant function analysis, and factor analysis. In addition to other tests of assumptions, it is particularly important to check for normality and multicollinearity. Some TAACO features may be particularly rare (or conversely particularly common) in certain datasets, leading to heavily skewed distributions. Furthermore, some TAACO indices measure the same construct in slightly different ways, which may result in a number of strongly correlated indices.

8. Interpret and Report Results

After conducting preliminary assumptions tests and (provided those assumptions are met) primary statistical tests are conducted, it is time to interpret the results. Interpreting the results should be (relatively) easy provided the researcher followed steps 1, 2, and 3. Researchers who neglect to follow these steps may find themselves having difficulty understanding and interpreting their results.