Neural Text Embeddings in Psychological Research: A Guide With Examples in R

Louis Teitelbaum and Almog Simchon

Department of Psychology, Ben-Gurion University of the Negev

Author Note

Louis Teitelbaum https://orcid.org/0009-0001-9347-0145

Almog Simchon https://orcid.org/0000-0003-2629-2913

Correspondence concerning this article should be addressed to Louis Teitelbaum, Department of Psychology, Ben-Gurion University of the Negev, POB 653, Beer Sheva 84105, Israel, Email: louist@post.bgu.ac.il

Abstract

As the capabilities of neural language models grow, psychologists in various subfields have begun to use them as tools in analysis. We introduce leading methods for quantifying psychological dimensions of text using neural semantic embeddings, and discuss their respective advantages and disadvantages. For the uninitiated reader, we provide an intuitive conceptual introduction to semantic embeddings, and discuss practical and theoretical differences between embeddings generated by word embedding models (e.g. word2vec; GloVe) and those generated by transformer-based large language models (LLMS). We review three methods of quantifying psychological constructs in embedding spaces: Distributed Dictionary Representation (DDR), Contextualized Construct Representation (CCR), and Correlational Anchored Vectors (CAV). We explore potential pitfalls of each method, and recommend best practices for their application in research. For each method introduced, we provide sample code in R.

Keywords: natural-language-procressing, r, deep-learning, word-embeddings, text-embeddings

Neural Text Embeddings in Psychological Research: A Guide With Examples in R

Psychologists have long insisted that talk therapy can heal, and that questionnaires can accurately measure psychological phenomena. These are language-based techniques, which rely on the assumption that language processing is linked to more fundamental internal states. Even so, psychological research has historically been unable to study *naturalistic language*—the sort of language that people produce in their day-to-day lives.

The first barrier to research of naturalistic language is that it has historically been difficult to record. Early efforts by linguists to record language samples from representative populations were heroic; starting in 1896, Edmond Edmont spent four years traveling around France on a bicycle conducting specially designed interviews to collect data for the *Atlas linguistique de la France*. He collected data from 700 participants in total (Crystal, 1997). Since then, microphones have made it easier to record speech, and the advent of transformer neural networks has made it possible to accurately transcribe speech at minimal cost. While these advances in audio processing are important, the most important recent change in the availability of natural language to psychologists has come through a different medium: text. Only a few decades ago, public access to text was limited to highly edited long-form productions like books, magazines, and newspapers. Some psychologists studied diaries or personal letters (e.g. Allport, 1942; Creegan, 1944), but personal documents like these are hard to collect at scale. With the rise of social media, minimally edited, naturalistic text has become the norm. Now more than ever before, people communicate through text—not just in long-distance correspondence, but for day-to-day socializing with friends and family. Moreover, much of this textual communication is synchronous and shares many of the same features as face-to-face spoken conversation (Placiński & Żywiczyński, 2023). Most importantly, much of this textual communication is freely available to researchers through social media platforms like Reddit and YouTube.

A second barrier to the use of naturalistic language in psychology is the problem of quantification. Language is complex, with near-infinite ways to describe the same thing. There are no clear rules for measuring the extent to which a text reflects depression, anxiety, mania,

introspection, or any other psychological construct. Before the past two decades, the few researchers who tried to extract quantitative psychological dimensions of text were nearly as heroic as Edmond Edmont on his four year journey around France. For example, Peterson and Seligman (1984) administered a questionnaire that prompted participants to write short explanations of various hypothetical scenarios. They then carefully read each response, noted each time a phrase like "because of" or "as a result of" was mentioned, and marked the accompanying explanation. These explanations were then typed by hand and shown one at a time to four trained judges who rated them on various 7-point scales. Finally, the agreement between the judges was assessed and their ratings were aggregated into the final variable used in their analysis of risk factors for depression. Today, this sort of analysis could be performed in a matter of seconds using the methods discussed in this guide.

Nuanced quantitative measures of language can also solve a third historical problem with naturalistic language in psychology: its social nature. People do not write or speak in a vacuum, they participate in conversations or group discussions, considering their audiences as they form their words. For the researcher, this means that language is full of uncontrolled, confounding variables: Is the speaker responding to another speaker? Who is the other speaker? How many participants are there in the conversation? Researchers in the field of psycholinguistics have tried to solve these problems by isolating speakers in a laboratory setting, contriving situations in which participants process and produce speech without the uncontrolled variability of conversational partners (O'Connell & Kowal, 2003). Nevertheless, the inherently social nature of language has made it difficult to analyze language behavior in even remotely naturalistic settings. A few decades ago the question "How similar are Daniel's utterances to Amos's utterances?" would have seemed hopelessly ill-defined. Similar in what way? Today, answering this question is simple with vector-based text embeddings.

In this guide, we introduce state of the art methods for using neural text embeddings to quantify psychological dimensions of text. We aim to explain the conceptual foundations of these methods in a way that is accessible to psychological researchers with no background in machine

learning or mathematics. We note methodological concerns and give concrete recommendations regarding proper usage of the techniques discussed. Throughout, we provide example code in R using the *tidyverse* (Wickham et al., 2019) and *Quanteda* (Benoit et al., 2018) packages. Other packages are used more sparingly and will be noted as they become relevant.

```
library(tidyverse)

library(quanteda)
```

We assume that the reader has basic proficiency in both of these frameworks. For a broader introduction to psychological text analysis using the tidyverse and Quanteda, see Teitelbaum and Simchon (2024). To follow along with the example code for this guide, please also load our suite of custom functions, available on our Github repo.[1] For readers who are primarily interested in the example code, the Github repo offers full example workflows for each method discussed.

```
source("embedding_scripts.R")
```

Abstract Embedding Spaces

The modern ability to generate nuanced, quantitative representations of language relies on a key assumption: The meaning of any word (or other subcomponent of text, called a "token") can be represented as a point in a single, continuous space. This space is referred to as the *semantic embedding space*, since tokens are "embedded" into it according to their semantic content. Points close to each other in the embedding space have similar meanings, while those far away convey different meanings. While certain directions in this space may correspond to a familiar scale along which meaning can vary (e.g. largeness vs. smallness), the dimensions of the space are essentially abstract; points are only defined relative to each other, through the use of distance metrics. Embedding spaces can be difficult to imagine, not only because of their abstract nature but also because they have many more than three dimensions—often hundreds or even thousands. This high dimensionality is necessary to represent the many ways in which linguistic meaning can vary.

[1] https://github.com/rimonim/embeddings_tutorial/blob/master/embedding_scripts.R

Navigating Vector Spaces: Distance and Similarity

Once words or texts are embedded into the semantic space, how do we quantify their relative positions? Here we introduce three metrics by which two points in an embedding space can be compared: most Euclidean distance, cosine similarity, and the dot product. While these do not constitute a comprehensive review of available similarity metrics, we consider them sufficient for making effective use of neural text embeddings in psychological research.

Euclidean Distance

The most straightforward way to measure the similarity between two points in space is to measure the distance between them. Euclidean distance is the simplest sort of distance—the length of the shortest straight line between the two points. The Euclidean distance between two vectors A and B can be calculated in any number of dimensions n using the following formula:

$$d\left(A, B\right) = \sqrt{\sum_{i=1}^{n} \left(A_i - B_i\right)^2}$$

A low Euclidean distance means two vectors are very similar. Euclidean distance has the advantage of being familiar to many non-specialists, but it is not usually well suited to the organizational structure of neural text embeddings and can result in higher rates of false positives than other similarity metrics (see Appendix B).

Cosine Similarity

The most common way to measure the similarity between two embedding vectors is with cosine similarity. This is the cosine of the angle between the two vectors. Since the cosine of 0 is 1, a high cosine similarity (close to 1) means two vectors are very similar. To give a simplified example, we consider the vectors in Table **??**.

The cosine is convenient because it is always between -1 and 1: When the two vectors are pointing in a similar direction, the cosine is close to 1, and when they are pointing in a near-opposite direction (180°), the cosine is close to -1.

Looking at Figure **??**, a question arises: Why should the angle be fixed at the zero point?

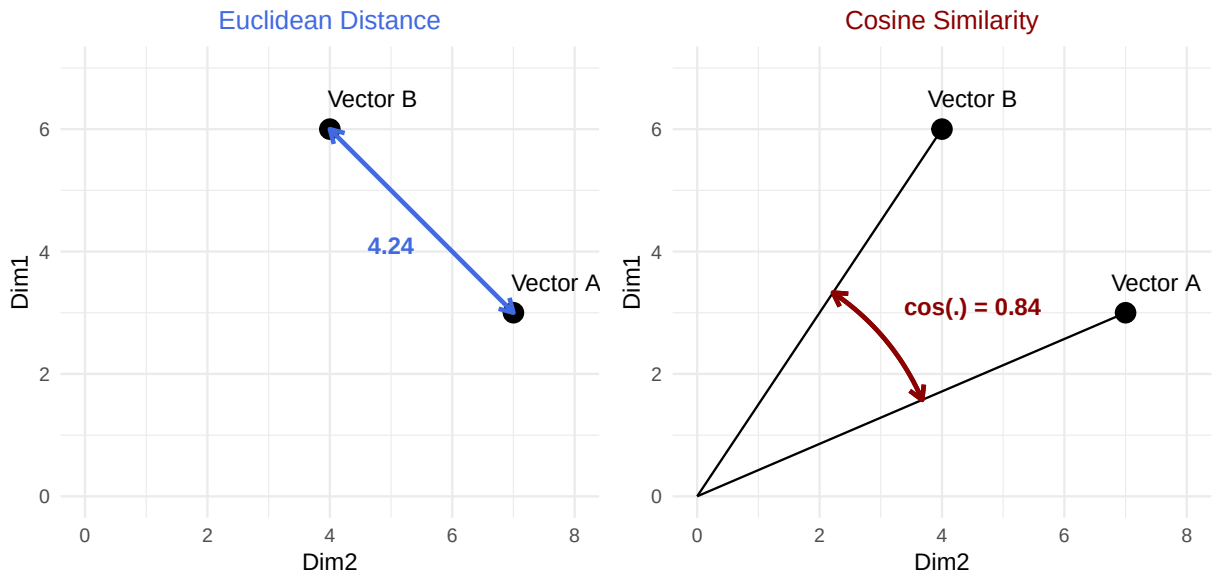
Table 1

Example Three-Dimensional Vectors

vector	Dim1	Dim2	Dim3
Vector A	3	7	2
Vector B	6	4	8

Figure 1

Two Simple Distance Measures



What does the zero point have to do with anything? Indeed, cosine similarity works best when the vector space is centered at zero (or close to it). In other words, it works best when zero represents a medium level of each variable. This fact is sometimes taken for granted because, in practice, many vector spaces are already centered at zero. In the case of neural word embeddings, as we shall see, this property is guaranteed by the model's architecture. In the case of transformer neural networks however, the use of specialized models may be necessary to guarantee zero-centered embedding spaces.

The formula for calculating cosine similarity is as follows:

$$Cosine(A, B) = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

In R, this can be translated to the following code:

```
#  - `x`: a numeric vector
#  - `y`: another numeric vector
cos_sim <- function(x, y){
  dot <- x %*% y
  normx <- sqrt(sum(x^2))
  normy <- sqrt(sum(y^2))
  as.vector( dot / (normx*normy) )
}
```

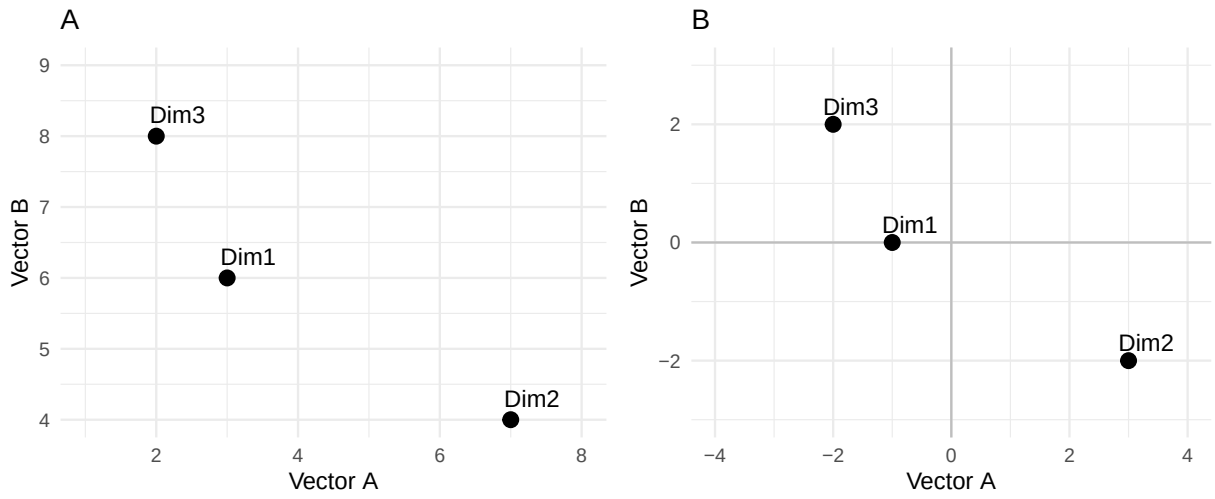
Readers comfortable with cosines may be satisfied with the explanation we have given so far. Nevertheless, many psychologists might find it helpful to consider the relationship between cosine similarity and a more familiar statistic that ranges between -1 and 1: the Pearson correlation coefficient. Cosine similarity measures the similarity between two *vectors*, while the correlation coefficient measures the similarity between two *variables*. Now just imagine our vectors as variables, with each dimension as an observation, as shown in Figure **??** A.

Now imagine centering those variables at zero, as shown in Figure **??** B. When seen like this, the correlation is the same as the cosine similarity. In other words, the correlation between two vectors is the same as the cosine similarity between them when the values of each vector are centered at zero.[2] Seeing cosine similarity as the non-centered version of correlation might provide additional intuition for why cosine similarity works best for vector spaces that are centered at zero.

[2] For a mathematical presentation of this relationship, see O'Connor (2012)

Figure 2

Vectors as Variables



Dot Product

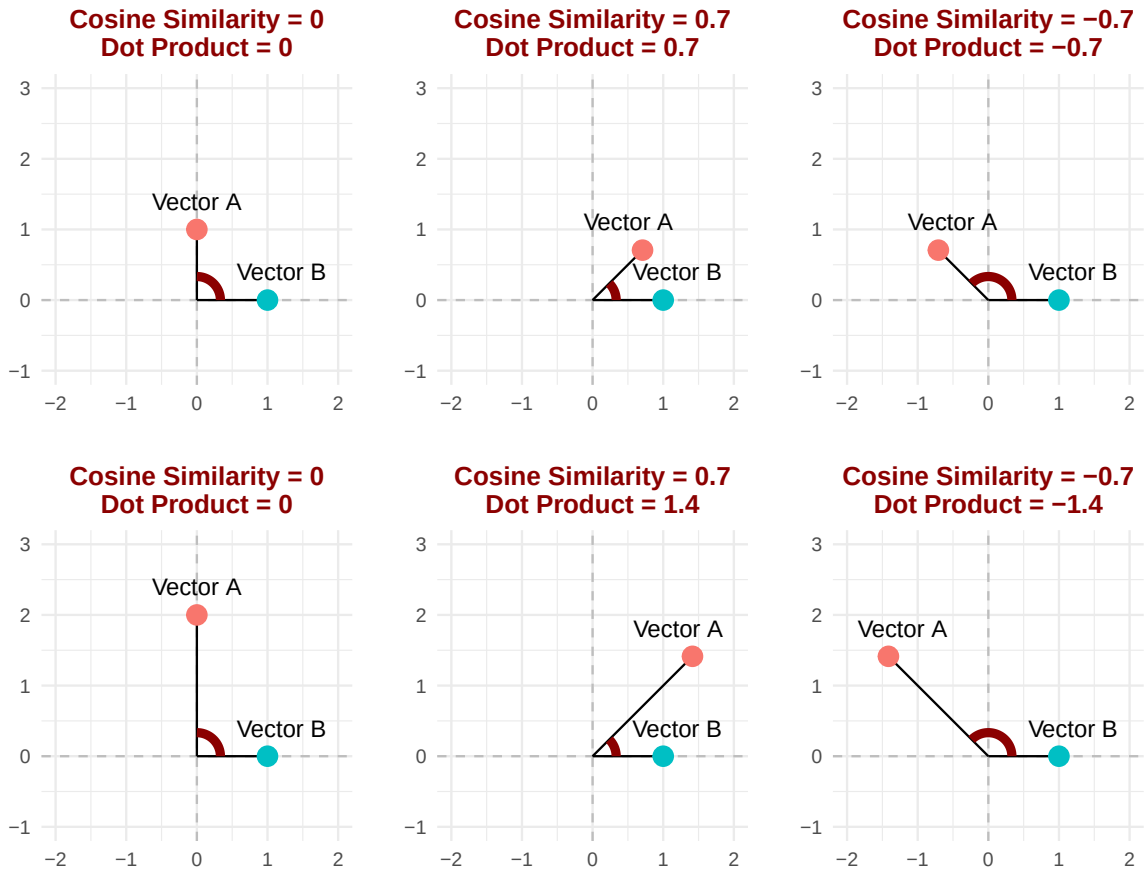
The dot product (sometimes called the inner product) of two vectors is similar to cosine similarity, with the exception that it gets larger as the vectors get further away from the origin (i.e. cosine similarity is the dot product of two normalized vectors).

```
#  - `x`: a numeric vector
#  - `y`: another numeric vector
dot_prod <- function(x, y){
  dot <- x %*% y
  as.vector(dot)
}
```

As we shall see later in this guide, the dot product is used internally in some neural embedding models to quantify the relationships between embeddings (e.g. Mikolov, Chen, et al., 2013; Pennington et al., 2014), providing the basis for much geometrical interpretation of embeddings produced by these models (Ethayarajh et al., 2019). We discuss the implications of

Figure 3

Dot Product vs. Cosine Similarity



this fact later in this guide, after introducing the models in question.

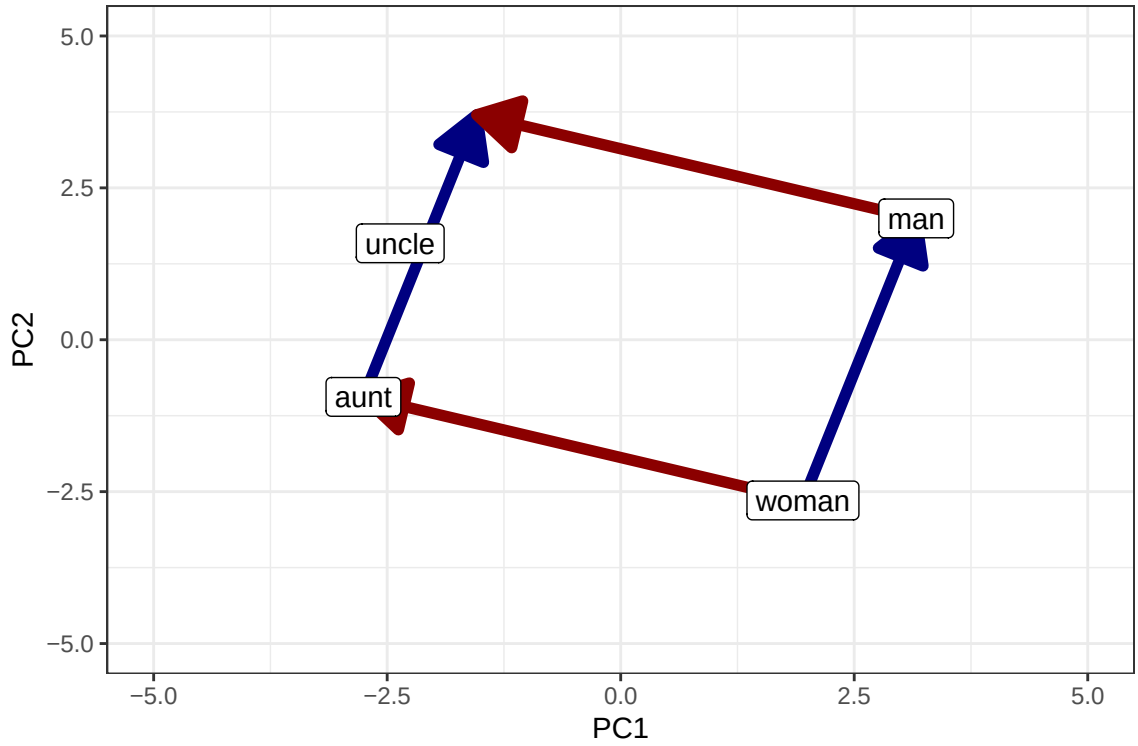
Additive Analogies and Anchored Vectors

An oft-touted property of embeddings from some models is that they can be added to each other in order to arrive at new concepts. A prototypical example is shown in Figure **??**. Subtracting the embedding of "man" from the embedding of "woman" results in the vector shown in blue. This vector represents the move from male to female gender. A vector between two embeddings is called an *anchored vector*. When the man-woman anchored vector is added to the embedding of "aunt", the result is very close to the embedding of "uncle". This property was first noted in word2vec (Mikolov, Yih, et al., 2013), and GloVe (Pennington et al., 2014) was

specifically designed with it in mind.

Figure 4

An Additive Analogy in Word Embeddings



Note. Word embeddings were acquired from the glove.twitter.27B.100d pretrained model and reduced to two dimensions with Principle Component Analysis (PCA).

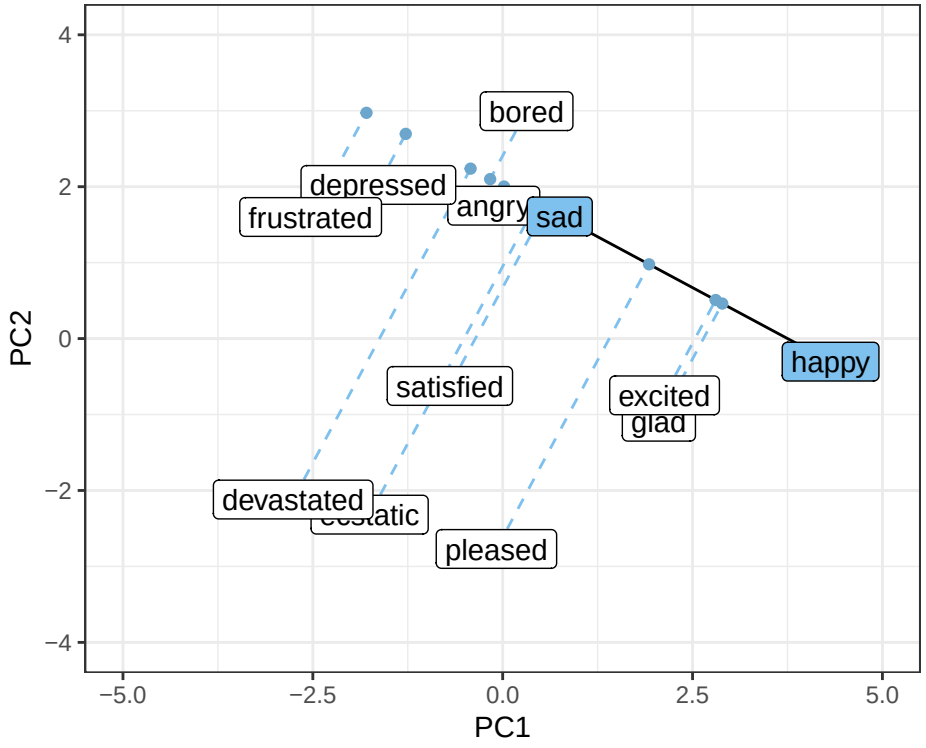
Additive analogical reasoning can help overcome a fundamental challenge of semantic embeddings. Consider the embeddings for "happy" and "sad". These may seem like opposites, but actually they are likely to be very close to each other in vector space because they both relate to emotional valence. This means that if we try to measure the happiness of words by comparing their embeddings to the embedding for "happy", we will actually be measuring the extent to which the words relate to emotion in general. The word "depression" might seem happier than the word "table", since depression is more emotion-related. This problem can be solved by using anchored vectors. Just as we created an anchored vector between "man" and "woman" to represent masculinity (as opposed to femininity) in Figure **??**, we can create an anchored vector between

"happy" and "sad" to represent happiness (as opposed to sadness). Such anchored vectors can be applied wherever necessary in embedding space.

To measure a construct with an anchored vector, take the dot product of a text embedding with the anchored vector. This is the equivalent of "projecting" the embeddings down onto the scale between one end of the anchored vector and the other, as shown in Figure **??**.[3]

Figure 5

Projection of Embeddings Onto an Anchored Vector



Note. Word embeddings were acquired from the glove.twitter.27B.100d pretrained model and reduced to two dimensions with Principle Component Analysis (PCA).

By projecting each embedding down onto the anchored vector between happy and sad, we

[3] For an intuitive explanation of why the dot product is equivalent to a projection, see 3blue1brown's video on the subject, at https://youtu.be/LyGKycYT2v0?si=86cfrN9DP9xw5HUx. Incidentally, the dot product with the anchored vector is also equivalent to the dot product with the positive embedding (e.g. "happy") minus the dot product with the negative vector (e.g. "sad").

create a scale from happy to sad.[4] This is sometimes referred to as **semantic projection** (Grand et al., 2022).

The Distributional Hypothesis

How do language models embed tokens in a semantic embedding space? By what metric is a word considered similar to or different from another? In answering this question, modern techniques rely on a further assumption, known as the distributional hypothesis. According to this assumption, tokens carry similar meanings inasmuch as they occur in similar contexts. For example, consider the following two sentences from the paper that introduced the distributional hypothesis, Harris (1954, emphasis added).

> "The formation of new *utterances* in the *language* is therefore based on the distributional relations as changeably perceived by the *speakers*-among the parts of the previously heard *utterances*."

> "The correlation between *language* and *meaning* is much greater when we consider connected discourse."

Even if we have no idea what "utterances" or "meaning" are, we can learn from these sentences that they must be related somehow, since they both appear together with the word "language." The more sentences we observe, the more sure we can be about the distributional patterns (i.e. which words tend to have similar words nearby). Words that tend to have very similar words nearby are likely to be similar in meaning, while words that have very different contexts are probably unrelated. Algorithms that learn the meanings of tokens (or at least the relations between their meanings) from these patterns of co-occurrence are called Distributional Semantic Models (DSMs).

---

[4] Taking the dot product with an anchored vector yields an unstandardized version of this scale. For a standardized version of this function, in which "sad" is at 0 and "happy" is at 1 on the scale, use the `anchored_sim()` function included in our Github repo at https://github.com/rimonim/embeddings_tutorial/blob/master/embedding_scripts.R.

A Common Misconception. Two words are NOT considered similar based on whether they appear together often. Words are similar when they tend to appear in similar *contexts*. For example, "fridge" and "refrigerator" almost never appear together in the same sentence, but they do tend to appear next to similar groupings of other words (e.g. "food," "cold," etc.).

Word Embeddings: word2vec, GloVe, and FastText

Word2vec

word2vec was first introduced by Mikolov, Chen, et al. (2013) and was refined by Mikolov, Sutskever, et al. (2013). They proposed a few variations on a simple neural network[5] that learns the relationships between words and contexts. Here we describe the most commonly used variation—continuous skip-gram with negative sampling.

Imagine training the model on the following sentence: "Coding is frustrating." The skip-gram training dataset would have one column for the input word, and another column for words from its immediate context. The technique is called "continuous" because it slides a context window along the training text, considering each word as input and the words immediately around it as context (e.g. 10 before and 10 after), as shown in Table **??**.

The negative sampling method adds more rows to the training set, this time from words and contexts that do not go together, drawn at random from other parts of the corpus. A third column indicates whether the pair of words are in fact neighbors or not, as shown in Table **??**.

The word2vec model takes the first two columns as input and tries to predict whether the two words are neighbors or not see **??**. It does this by learning two separate sets of embeddings: word embeddings and context embeddings. For each row of the training set, the model looks up the embedding for the target word and the embedding for the context word, and computes the dot product between the two vectors. If the dot product is large (i.e. the word embedding and the context embedding are very similar), the model predicts that the two words are likely to be real

[5] Some experts consider word2vec too simple to be called a neural network. The precise definition of "neural network" is beyond the scope of this guide; word2vec could just as productively be thought of as an advanced form of logistic regression.

Table 3

Example Skipgram Training Set

word	context
coding	is
coding	frustrating
is	coding
is	frustrating
frustrating	coding
frustrating	is

Table 5

Example Skipgram Training Set With Negative Sampling

word	context	neighbors
coding	is	1
coding	frustrating	1
is	coding	1
is	frustrating	1
coding	happy	0
coding	olive	0
is	happy	0
is	olive	0

neighbors. If the dot product is small, the model predicts that the two words were probably sampled at random.[6] During training, the model learns which word embeddings and context

[6] To learn why models like word2vec use dot products instead of cosine similarity, see "Interpreting Word Embeddings" below.

embeddings will do best at this binary prediction task.

Note that word2vec (as well as fastText and GloVe) give each word two embeddings: one for when the word is the target and another for when it is the context (Goldberg & Levy, 2014). This overcomes two important challenges of semantic embedding:

1. **A Nuance of the Distributional Hypothesis.** Recall the case of "fridge" and "refrigerator", which almost never appear together in the same sentence, but do tend to appear next to similar groupings of other words. Earlier word embedding models such as latent semantic analysis (Deerwester et al., 1990), which are based directly on broad patterns of covariance in word frequencies, will pick up on the fact that "fridge" and "refrigerator" are negatively correlated and push them further apart than they should be. Word2vec, on the other hand, can learn a *context embedding* for "refrigerator" that is not so close to the *word embedding* for "fridge", even when the word embeddings of the two words are very close. This allows word2vec to recognize that "refrigerator" and "fridge" tend to appear in similar contexts, but are unlikely to appear together.
2. **Associative Asymmetry.** The cosine similarity between two word embeddings gives the best estimate of *conceptual similarity* (Torabi Asr et al., 2018). This is because conceptual similarity is not the same as association in language (or in the mind). In fact, psycholinguists have long known that human associations between two words are asymmetric. For example, people prompted with "leopard" are much more likely to think of "tiger" than people prompted with "tiger" are to think of "leopard" (Tversky & Gati, 1982). These sorts of associative connections are closely tied to probabilities of co-occurrence in language and are therefore much better represented by the cosine similarity (or even the dot product) between a word embedding and a context embedding (Torabi Asr et al., 2018). Thus the association between "leopard" and "tiger" would be represented by the similarity between the *word embedding* of "leopard" and the *context embedding* of "tiger", allowing for the asymmetry observed in mental associations.[7] Since older models like latent semantic

[7] To the best of our knowledge, pretrained context embeddings are not available online. For associative (rather than

analysis only produce one embedding per word, they cannot capture this asymmetry.

Some pretrained word2vec models can be easily downloaded from the Internet[8]. Because these models are trained on very large datasets and are already known to perform well, it not necessary to train a new word2vec from scratch.[9]

A downloaded pretrained model (generally a .bin file), can be opened in R with the *word2vec* package.[10] In the example code below, we use a model trained on Google news, which uses 300-dimensional embeddings.[11]

```
library(word2vec)


# model file path
word2vec_mod <- "data/GoogleNews-vectors-negative300.bin"


# open model
word2vec_mod <- read.word2vec(file = word2vec_mod, normalize = TRUE)
```

To find embeddings of specific words using the *word2vec* package, use

---

conceptual) relationships between words, we recommend training a new model. A guide on training a custom GloVe model in R can be found at https://quanteda.io/articles/pkgdown/replication/text2vec.html.

[8] See for example https://github.com/maxoodf/word2vec?tab=readme-ov-file#basic-usage and https://www.kaggle.com/datasets/pkugoodspeed/nlpword2vecembeddingspretrained

[9] Training custom word embeddings may be useful in certain specialized cases. For example, a researcher who is interested in quantifying differences in individual word use between multiple large groups of text might train a GloVe model on texts written by conservatives and another on texts written by liberals, and demonstrate that the word "skirt" is closer to the word "woman" in conservative language than it is in liberal language. A guide on training custom GloVe models in R can be found at https://quanteda.io/articles/pkgdown/replication/text2vec.html.

[10] Available at https://cran.r-project.org/web/packages/word2vec/readme/README.html

[11] Available at https://www.kaggle.com/datasets/pkugoodspeed/nlpword2vecembeddingspretrained/download?datasetVersionNumber=1

`predict(word2vec_mod, c("word1", "word2"), type = "embedding")`. To get embeddings for full documents, average the embeddings of the words in the document. In our github repo, we provide the `textstat_embedding` to compute document embeddings directly from a document-feature matrix (DFM).[12]

Word2vec: Key Takeaways. By distinguishing between target and context words, word2vec allows greater fidelity to the distributional hypothesis. Since it is not based on counts, it also avoids problems with non-linear relationships and irregular distributional properties of word frequencies (see Baayen, 2001). Word2vec, like other word embedding models, is also efficient for use on large datasets, since pretrained embeddings can be looked up for each word. The main disadvantage of word2vec is that it assumes that each word has only one meaning. This means that it has trouble with words that can mean more than one thing (e.g. deep learning *model* vs. fashion *model*). Word2vec will learn the average of these meanings. Furthermore, word2vec is primarily effective in English. This is because English words are generally spelled the same no matter where they are in a sentence. Word2vec does not work as well for languages that have more prefixes, suffixes, conjugations, etc., since it has to relearn the meaning for each form of the word. Finally, word2vec is made less appealing to researchers due to the relative paucity of pretrained models available online.

GloVe

Word2vec produces spectacularly rich and reliable vector embeddings, but their reliance on randomly sampled pairs of words and contexts makes them somewhat noisy and overly sensitive to frequent tokens. The developers of word2vec managed to fix these problems by strategically filtering the training dataset. Pennington et al. (2014), in contrast, developed a refactored model that corrects for frequency internally: Global Vectors (GloVe) is designed on the same principles as word2vec, but it is computed from global patterns of co-occurrence rather than individual examples.

Although GloVe uses a different method of training, the embeddings it generates are very

[12] https://github.com/rimonim/embeddings_tutorial/blob/master/embedding_scripts.R

similar to those generated by word2vec. Because GloVe embeddings are so similar to word2vec embeddings, we will not go into detail here about the way the GloVe algorithm works. Nevertheless, GloVe does have one very important advantage over word2vec: Better pretrained models are available online. Whereas the most easily available word2vec model is trained on news articles, the GloVe website[13] offers models trained on social media (`glove.twitter.27B.zip`) and on large portions of the Internet (Rana, 2010). These models generalize better to social media texts (since they were trained on similar texts) and are likely to have richer representations of emotional or social content, since more examples of that content appear on social media than in the news or on Wikipedia.[14]

Since the pretrained GloVe models are available in .txt format, a wrapper package is not required to use them in R. The following code loads a downloaded pretrained model for use.

```
path_to_glove <- "~/Projects/ds4psych/data/glove/glove.twitter.27B.100d.txt"
glove_pretrained <- load_embeddings_txt(path_to_glove)
```

GloVe: Key Takeaways. GloVe's efficient training procedure has made it a popular choice for researchers who wish to train their own models. GloVe is also appealing for researchers looking for high quality, psychologically sensitive, pretrained models, since a number of such models are available for download online. Like word2vec, GloVe's primary downsides are that it relies on word-level meaning and that it has limited applicability to languages other than English.

FastText

FastText (Bojanowski et al., 2017) is a specialized version of word2vec, designed to work with languages in which words take different forms depending on their grammatical place. Rather

[13] https://nlp.stanford.edu/projects/glove/

[14] Another notable difference between GloVe and word2vec is that the GLoVe averages the word embeddings and context embeddings rather than using only the word embeddings as word2vec does. This makes GloVe embeddings slightly better at representing overall meaning, but may blur the distinction between conceptual similarity and mental/linguistic association (Torabi Asr et al., 2018).

than learning a word embedding and a context embedding for each full word (e.g. “quantify” and “quantification” each get their own embedding), fastText learns a vector for each string of characters within a word (such a string is sometimes referred to as a shingle). For example, “quantify” might be broken up into “quant”, “uanti”, “antif”, and “ntify”. But it doesn’t treat each shingle as its own word. Rather, it trains on words just like word2vec and GloVe, but makes sure that the embedding of a word is equal to the *sum* of all of the shingle vectors inside it.

This approach is mostly unnecessary for English, where words are generally spelled the same wherever they appear. But for more morphologically rich languages like Hebrew, Arabic, French, or Finnish, fastText works much better than word2vec and GloVe. This is because there might not be enough data for word2vec and GloVe to learn reliable representations of every form of every word, especially rare forms. FastText, on the other hand, can focus on the important subcomponents of the words that stay the same across different forms. This way it can learn rich representations even of rare forms of a word that don’t appear in the training dataset (e.g. it could quantify the meaning of “morphologically” even if it were only trained on “morphology”, “morpheme”, and “chronologically”).

After downloading a pretrained model from Grave et al. (2018), you can use fastText in R through the *fastTextR* package.[15] *FastTextR* includes a dedicated function for obtaining full text embeddings, `ft_sentence_vectors()`.

```
library(fastTextR)


# example French texts
fr_words <- c("Français", "française")
fr_texts <- c("Ce qui n'est pas clair n'est pas français.",
              "Ce que l'on conçoit bien s'énonce clairement")
```

[15] Available at https://cran.r-project.org/web/packages/fastTextR/vignettes/Word_representations.html

```
# load pretrained model from file
fr_model <- ft_load("data/cc.fr.300.bin")


# get word embeddings
word_vecs <- ft_word_vectors(fr_model, fr_words)


# get text embeddings
text_vecs <- ft_sentence_vectors(fr_model, fr_texts)
```

FastText: Key Takeaways. Relative to word2vec and GloVe, fastText performs much better on morphologically rich languages. Pretrained models for 157 languages are available online.[16] Furthermore, fastText has increased performance for rare words, and is able to infer embeddings for words that were not in the training data. The cost of this gain is that fastText models are more complex, resulting in larger files to download when using pretrained models. The added complexity may also increase the risk of overfitting.

Interpreting Word Embeddings

Advanced word embedding algorithms like word2vec, GloVe, and fastText use the dot product of two word embeddings to measure how likely the probability that the words occur together. Vectors that are farther away from the origin will result in very positive or very negative dot products, making the model more confident in the pair of words either being neighbors or not. This means that the distance of a word embedding from the origin (also called the norm or magnitude) is proportional to the informativeness of the word (Oyama et al., 2023; Schakel & Wilson, 2015). For example, the word "the" has a very low magnitude because it does not indicate a specific context, while the word "psychology" has a very high magnitude because its use is associated with a very specific context. Therefore, the magnitude of the embedding measures how representative it is of certain contexts as opposed to others.

[16] https://fasttext.cc/docs/en/crawl-vectors.html

This is the reason why an accurate embedding of a full text can be obtained by averaging the embeddings of each of its words. One may assume that averaging word embeddings will lead to overvaluing common words, like “the” and “I”, which appear more frequently but are not very informative about the text’s meaning. In fact, however, the magnitude of a word embedding is smaller for common words, which means that common words have less impact on the average (Ethayarajh et al., 2019).

Once average embeddings are computed, most researchers use cosine similarity to assess the relationships between embeddings. The cosine similarity measures only the meanings of the two embeddings, while ignoring how specific they are to those meanings. If the specificity of texts to a construct of interest is important to the analysis, the dot product may be more appropriate than the cosine similarity. For this reason, the dot product may sometimes be optimal for analyzing texts with decontextualized embeddings (See Appendix A).

Contextualization With Large Language Models

The models we have discussed so far represent the meaning of each token as a point in multidimensional space: a word embedding. Word embeddings generated by models like word2vec or GloVe are often referred to as **decontextualized embeddings**. This name may be confusing, since the purpose of these models is to associate tokens with the contexts in which they tend to appear. Nevertheless, these models are decontextualized in that they can allow only one context representation per token—they therefore represent the average of the contexts in which a token appears throughout the training corpus. For example, consider the following uses of the token “short”.

My dad is very *short*.

My blender broke because of a *short* circuit.

That video was anything but *short*.

I can’t pay because I’m *short* on cash at the moment.

Any speaker of English can easily see that the word "short" means something different in each one of these examples. But because word2vec and similar models are trained to predict the context based on only a single word at a time, their representation of the word *short* will only capture that word's average meaning.

How can we move beyond the average meaning and capture the different meanings words take on in different contexts? Readers may be familiar with Large Language Models (LLMs) like ChatGPT and Claude. At their core, much of what these models do is exactly this: They find the intricate relationships between tokens in a text and use them to develop a new understanding of what these tokens mean in the particular context of that text. For this reason, embeddings produced by these models are often referred to as **contextualized embeddings**.

The core of all modern LLMs is a model called the *transformer* (Vaswani et al., 2017). We will not cover exactly how transformers work. For the purposes of this guide, we offer only the broad overview necessary to make proper use of them in research: In processing a text, transformers first convert all the words into word embeddings, as word2vec or GloVe would do. At the start, these word embeddings represent the average meaning of each word. The transformer then estimates how each word in the text might be relevant for better understanding the meaning of the other words. For example, if "circuit" appears immediately following "short", the embedding of "short" should probably be tweaked. Once it has identified this connection, the transformer computes what "circuit" should add to a word that it is associated with, moving the "short" embedding closer to embeddings for electrical concepts. A full LLM has many *layers*. In each layer, the LLM identifies more connections between embeddings and shifts the embeddings in the vector space to add more nuance to their representations. When it gets to the final layer, the LLM uses the enriched embeddings of the words in the text for whatever task it was trained to do (e.g. predicting what the next word will be, or identifying whether the text is spam or not).

In order to extract an LLM's rich, contextualized embeddings, researchers can run it on a text and stop it before it finishes predicting the next word (or anything else it was trained to do). This way, we can read the LLM's mind, capturing all of the rich associations it has with the text.

Hugging Face and the *text* Package

Leading commercial LLMs like GPT-4 are hidden behind APIs so that their inner workings are kept secret. Researchers therefore cannot access the embeddings of these high profile models. Nevertheless, plenty of models that are almost as good are open source and easily accessible through Hugging Face Transformers.[17] Any text-based transformer model can be accessed in R using the *text* package (Kjell et al., 2021).[18]

The key tool provided by the *text* package is `textEmbed()`. This function takes in raw texts and generates contextualized embeddings using a specified open source model. Here we use `BAAI/bge-base-en-v1.5` (Xiao et al., 2023), a relatively lightweight model that nonetheless achieves state-of-the-art performance on the MTEB benchmark, a leading resource for evaluating text embedding models (Muennighoff et al., 2022).[19] By default, `textEmbed()` creates the full text embedding by averaging the contextualized embeddings of each token in the text. However, `BAAI/bge-base-en-v1.5` is trained to provide optimal text embeddings in the final layer embedding of the `[CLS]` token in particular.[20] When using `BAAI/bge-base-en-v1.5` to generate embeddings, it is therefore necessary to specify `tokens_select = "[CLS]"` and `layers = -1`. This has the added benefit of speeding up the computation.

---

[17] https://huggingface.co/docs/transformers/index

[18] The *text* package runs Python code behind the scenes, so new users will have to set up a Python environment for it to run properly. For instructions on how to do this, see https://www.r-text.org/articles/huggingface_in_r_extended_installation_guide.html.

[19] The current MTEB leaderboard can be found at https://huggingface.co/spaces/mteb/leaderboard.

[20] The `[CLS]` token is a special token that models based on BERT (Devlin et al., 2019) add to each text. Because the `[CLS]` token does not have a "real" meaning, but rather is inserted at the same place in every text, its contextualized embedding represents the gist of each text as a whole. In training, traditional BERT models use the contextualized embedding of the `[CLS]` token to predict whether a given text does or does not come after the input text. `BAAI/bge-base-en-v1.5` has a somewhat different training paradigm, but it maintains the conventional `[CLS]` token.

```r
library(text)


example_texts <- c(
  "This is one text.",
  "This is another."
  )


# run model
example_texts_bge <- textEmbed(
  example_texts,
  model = "BAAI/bge-base-en-v1.5", # model name
  tokens_select = "[CLS]", # select only [CLS] token embedding
  layers = -1  # last layer
)


# extract dataframe of embeddings
example_texts_bge <- example_texts_bge$texts[[1]]
```

Managing Computational Load

Running large neural networks on a personal computer can be time consuming. The following are some ways to lessen the computational load when calling `textEmbed()`:

- Use a smaller model. Hugging Face model pages generally state how many parameters the model has. This is a good indication of how much computational capacity the model needs. For example, consider using `distilbert-base-uncased` (67M params) or `albert-base-v2` (11.8M params) instead of `bert-base-uncased` (110M params).
- Only retrieve one layer at a time (e.g. `layers = -2`).
- If individual token embeddings are not needed, set `keep_token_embeddings = FALSE`.

- To avoid aggregation costs, only querry individual token embeddings (e.g. `tokens_select = "[CLS]"`).
- Run the model on the GPU with `device = 'gpu'` (not available for Apple M1 and M2 chips).
- Break up the dataset into smaller groups of texts, run each on its own, and join them back together afterward.

Before running `textEmbed()` on a full dataset, always try running it on two or three texts first. This will give a sense of how long the full computation will take, and will ensure that the output is as expected.

Choosing the Right Model

Part of the beauty of LLMs is their ability to generalize—most popular models today are trained on enormous datasets with a wide variety of content, and even smaller models perform more than well enough to capture straightforward psychological concepts like emotional valence (Kjell et al., 2022). Nevertheless, there is a notable major limitation of LLM embeddings. Word2vec, GloVe, and related models have architectures that guarantee vector spaces with consistent geometric properties, allowing researchers to confidently compute averages between vectors and interpret linear directions as encoding unique semantic meanings. In contrast, the ways that LLMs organize meaning in their embedding spaces are not well understood and may not always lend themselves to simple measures like those described in this guide [Cai et al. (2021); Li et al. (2020); Reif et al. (2019); zhou_etal_2022].

Recently, researchers have tried to fix this problem by creating models that are trained explicitly to produce high quality embeddings. These models encourage embeddings to spread out evenly in the embedding space, or explicitly optimize for reliable cosine similarity metrics. Many of these models can be found on the leaderboard for the MTEB benchmark (Muennighoff et al., 2022).[21]

[21] The current MTEB leaderboard can be found at https://huggingface.co/spaces/mteb/leaderboard.

Additive Analogies in Contextualized Embeddings

Note in Figure **??** that the analogical property relies on the magnitude of the vectors—if some vectors were shorter or longer than necessary, the parallelogram would not fit. This means that analogical reasoning may not be applicable to LLM embeddings, which are often organized in nonlinear patterns (Cai et al., 2021; Ethayarajh, 2019; Gao et al., 2019). Even specialized models like sentence transformers are generally not designed with the additive analogical property in mind (Reimers & Gurevych, 2019). Even though the geometrically motivated methods described in this guide work fairly well in LLM embeddings, there is room for improvement in this area.[22]

Contextualized Embeddings: Key Takeaways

Contextualized embeddings by definition can represent multiple senses of each word. This is especially important for analyzing shorter texts—since the sample of words is small, the need to correctly characterize each one is greater. Contextualized embeddings are also sensitive to word order and negation (e.g. LLMs can tell the difference between "he's funny but not smart" and "he's smart but not funny"). Despite these advantages, LLM embeddings come with two downsides. First, they are computationally expensive to generate, making them unfeasible for application to large datasets. Second, unlike simple word embedding models, LLMs may sometimes organize embeddings in nonlinear patterns. For example, GPT models tend to arrange their embeddings in a spiral (Cai et al., 2021). This is presumably why BERT text embeddings perform worse than word2vec and GloVe when analyzed using cosine similarity (Reimers & Gurevych, 2019). Specialized models such as those listed on the leaderboard for the MTEB benchmark

[22] Some promising methods have been suggested for extracting geometrically regular embeddings out of non-specialized LLMs. For example, averaging the last two layers of the model seems to help (Li et al., 2020). Taking a different approach, Ethayarajh (2019) created static word embeddings from an LLM by running it on a large corpus and taking the set of each word's contextualized representations from all the places it appears in the corpus. The loadings of the first principal component of this set represent the dimensions along which the meaning of the word changes across different contexts. These loadings can themselves be used as a vector embedding which can out-perform GloVe and FastText embeddings on many word vector benchmarks, including analogy solving. This approach worked best for embeddings from the early layers of the LLM.

(Muennighoff et al., 2022) can help with this problem, but are unlikely to solve it entirely.

Representing Psychological Constructs: DDR, CCR, and CAV

With modern vector-based methods, any language-based psychological measure can be represented as a vector. Psychology has used language-based measures like dictionaries and questionnaires for over a century. To smoothly continue this existing research in the age of vector spaces, we introduce three methods for quantifying existing psychological constructs in high-dimensional vector space: Distributed Dictionary Representation (DDR), Contextualized Construct Representation (CCR), and Correlational Anchored Vectors (CAV). As we introduce each method, we use it to quantify an example psychological construct: either self-referential processing or the related concept of locus of control—the feeling that one has control over one's own life (Rotter, 1966). Finally, we use each metric to test for significant differences between posts in the Reddit r/depression community (N = 995) and posts in the r/TodayIamHappy community (N = 781). As their names suggest, the former is a support group for users struggling with depression, while the latter is a forum for sharing happy experiences. This small sample of posts, which we name `reddit_emotion`, was collected from the Pushshift Reddit archives (Baumgartner et al., 2020) and is used here purely for the sake of demonstration.

Distributed Dictionary Representation (DDR)

We begin with a straightforward sort of psychological measure—the dictionary (also known as a word list or lexicon). A dictionary is a list of words (or other units of text, generally called "tokens") associated with a given psychological or other construct. For example, a dictionary for depression might include words like "sleepy" and "down." Psychological researchers have been constructing, validating, and publicizing dictionaries for decades. But these dictionaries are generally designed to be used by counting dictionary words in a text—How can they be applied to a vector-based analysis? Garten et al. (2018) propose a simple solution: Generate word embeddings for each word in the dictionary, and average them together to create a single DDR. The dictionary construct can then be measured by comparing text embeddings to the DDR.

DDR cannot entirely replace word counts; for linguistic concepts like conjunctions or use of the passive voice, word counts may still be psychometrically optimal. But DDR is ideal for studies of abstract constructs like emotions, that refer to the general gist of a text rather than particular words. The rich representation of word embeddings allows DDR to capture even the subtlest associations between words and constructs, and to precisely reflect the *extent* to which each word is associated with each construct. It can do this even for texts that do not contain any dictionary words. Because embeddings are continuous and already calibrated to the probabilities of word use in language, DDR also avoids the difficult statistical problems that arise due to the strange distributions of word counts (see Teitelbaum & Simchon, 2024, Chapter 16).

Garten et al. (2018) found that DDR works best with smaller dictionaries of only the words most directly connected to the construct being measured (around 30 words worked best in their experiments). Word embeddings work by overvaluing informative words—a desirable property for raw texts, in which uninformative words tend to be very frequent.[23] But dictionaries only include one of each word. In longer dictionaries with more infrequent, tangentially connected words, averaging word embeddings will therefore overvalue those infrequent words and skew the DDR. This can be fixed with Garten et al.'s method of picking out only the most informative words. Alternatively, it could be fixed by measuring the frequency of each dictionary word in a corpus and weighting the average embedding by that frequency. This method is actually more consistent with the way most dictionaries are validated, by counting the frequencies of dictionary words in text (for preliminary empirical validation of this method, see Appendix A).

In the example code below, we create a DDR for self-referential processing using the Linguistic Inquiry and Word Count (LIWC) dictionary of first person singular pronouns (Boyd et al., 2022). Since LIWC dictionaries include stems in addition to full words, we first expand the dictionary by running the LIWC software on each unique word in `reddit_emotion` (full code is

---

[23] For more information on this property, see "Interpreting Advanced Word Embeddings" above. Note that this property emerges naturally from the way decontextualized models like word2vec and GloVe are trained, and therefore may not hold true for contextualized embeddings.

available in our Github repo). This results in a character vector of dictionary words, called `i_dict` in the code below. To correct for word informativeness, we weight the dictionary word embeddings by their frequency in `reddit_emotion`.

```
# Embed Dictionary Words
i_dict_glove <- i_dict |>
  tokens() |>
  dfm() |>
  textstat_embedding(glove_pretrained) |>
  select(-doc_id) |>
  as.matrix()
rownames(i_dict_glove) <- i_dict


# Get Dictionary Word Frequencies
i_dict_freqs <- reddit_emotion_dfm |>
  dfm_keep(i_dict) |>
  quanteda.textstats::textstat_frequency() |>
  pull(frequency, name = feature)


# Aggregate dictionary embeddings with weighted average
i_DDR <- apply(i_dict_glove, 2, weighted.mean, w = i_dict_freqs)
```

To measure the similarity of Reddit posts to this theoretical construct, we now need only to compute the cosine similarity between the DDR and the embedding of each text. Beginning with a DFM of `reddit_emotion`, the following code performs this computation and conducts a t-test for differences between r/depression and r/TodayIamHappy.

```
# Embed Texts of Interest
reddit_emotion_glove <- reddit_emotion_dfm |>
  textstat_embedding(glove_pretrained) |>
  select(-doc_id)

# Calculate Distance Metrics
reddit_emotion_scores <- reddit_emotion_glove |>
  rowwise() |>
  mutate(selfref = cos_sim(c_across(V1:V100), i_DDR)) |>
  pull(selfref)

# Test Hypothesis
test_DDR <- t.test(
  x = reddit_emotion_scores[reddit_emotion$subreddit=="depression"],
  y = reddit_emotion_scores[reddit_emotion$subreddit=="TodayIamHappy"]
  )
test_DDR$p.value
```

```
[1] 1.453378e-07
```

```
test_DDR$estimate
```

```
mean of x mean of y
0.9266877 0.9195364
```

We find that posts in r/depression have more self-referential language ($p < .001$). This is consistent with existing research indicating that the use of self-referential language increases during depressive episodes (Rude et al., 2004).

DDR: Key Takeaways

DDR produces richer, more robust construct representation than traditional word counting methods (cf. Boyd et al., 2022). Since DDR representations are continuous, they also avoid statistical problems associated with the distribution of word counts (see Baayen, 2001). Another advantage of DDR over dictionary-based word counts is that DDR enables word-by-word analysis of text. It is not very informative to count how many dictionary words are in each word (it will either be one or zero), but the embedding of each word can be compared to the DDR—how close are they in the vector space? This allows researchers to see how a construct spreads out within a single text. Relatedly, DDR only needs a dictionary that captures the essence of the construct being measured. For many constructs, this could be only a few words. This property can allow researchers to create rich construct representations for novel constructs without investing in the development of a large dictionary. One downside of DDR is that it can implicitly encode associated constructs. For example, if surprised texts tend to have positive valence in the data used to train the word embedding model, the DDR for surprise may embed some positive valence as well. This can be remedied by constructing a DDR for positive valence too, and using it as a statistical control when testing hypotheses. Lastly, DDR may not be appropriate for linguistic measures. Word embeddings encode the general gist of a text, whereas constructs like passive voice or pronoun use refer to specific words. Constructs that are concretely tied to words or grammatical forms in the text should therefore be analyzed using standard word counting methods (see Boyd et al., 2022).

Contextualized Construct Representation (CCR)

Dictionaries are not the only validated psychological measures that we can apply using embeddings. With contextualized embeddings, we can extract the gist of any text and compare it to that of any other text. Atari et al. (2023) propose to do this with the most popular form of psychometric scale: the questionnaire. Psychologists have been using questionnaires to measure things for over a century, and tens of thousands of validated questionnaires are now available online. The LLM embedding of a questionnaire is referred to as a CCR.

We can use CCR to measure internal locus of control by using the questionnaire introduced by Rotter (1966). The questionnaire includes items measuring an internal locus of control (e.g. "What happens to me is my own doing"), and items measuring an external locus of control (e.g. "Sometimes I feel that I don't have enough control over the direction my life is taking"). In the code below, we compute a CCR for the internal locus of control items.[24]

```
# Embed Questionnaire Items
internal_bge <- text::textEmbed(
  internal_items,
  model = "BAAI/bge-base-en-v1.5", # model name
  tokens_select = "[CLS]", # select only [CLS] token embedding
  layers = -1,  # last layer
  dim_name = FALSE,
  keep_token_embeddings = FALSE
)
internal_bge <- internal_bge$texts[[1]]


# Aggregate Questionnaire Embeddings
internal_bge <- apply(as.matrix(internal_bge), 2, mean)
```

The ubiquity of questionnaire measures in psychological research makes CCR a powerful tool. Nevertheless, researchers should keep in mind what is being represented by the CCR. Embeddings capture the overall feel of a text, including its tone and dialect. With CCR, we are comparing the feel of a questionnaire written by academics to the feel of posts written by Reddit users. By comparing these vectors, we are not just measuring how much self-referential language is in each text—we are also measuring the extent to which each text is in the style of a

[24] Many questionnaires include reverse-coded items (e.g. "I often feel happy" on a depression questionnaire). The easiest way to deal with these is to manually add negations to flip their meaning (e.g. "I *do not* often feel happy").

questionnaire written by academics. This has the potential to introduce a confounding variable into the analysis—questionnaire-ness.

The questionnaire-ness problem suggests that CCR is most effective for analyzing texts that bear a strong similarity to the questionnaire itself. For example, the texts of interest may be participant descriptions of their own values, while the questionnaire items are statements about values in the first person (as is commonly the case). In cases like this, CCR is likely to perform optimally. With this method, researchers can compare participant responses to the questionnaire without actually administering the questionnaire itself; participants can answer in their own words, which CCR will compare to the wording of the questionnaire (for preliminary empirical validation of this recommendation, see Appendix B).

Anchored Vectors in CCR and DDR

A second potential remedy for the questionnaire-ness problem is to compute CCRs for both the construct of interest and its opposite, and to operationalize the construct as the anchored vector between them. In the code below, we compute an anchored vector for locus of control by subtracting the CCR of the internal locus of control items from that of the external locus of control items. This anchored vector then measures internal *as opposed to external* locus of control.

```
# Anchored vector

loc_anchored <- internal_bge - external_bge
```

Anchored vectors can be applied to any construct that has a clear opposite, whether operationalized through CCR, DDR, or other methods. Although their application to contextualized embeddings relies on strong assumptions about the linearity of the contextualized embedding space, they have been shown to work reasonably well for a variety of constructs and models (Grand et al., 2022). For an empirical demonstration of the robustness of anchored vectors for DDR, see Appendix A. For an empirical demonstration of their applicability to CCR, see Appendix B.

In the following code, we use the anchored vector for locus of control that we computed

above (`loc_anchored`) to measure locus of control in `reddit_emotions` and to test the hypothesis that posts in r/depression exhibit more external locus of control than those in r/TodayIamHappy (cf. Simchon et al., 2023).

```
# Calculate Dot Product With Anchored Vector
reddit_emotion_loc <- reddit_emotion_bge |>
  rowwise() |>
  mutate(loc = dot_prod(c_across(Dim1:Dim768), loc_anchored)) |>
  pull(loc)


# Test Hypothesis
test_CCR <- t.test(
  x = reddit_emotion_loc[reddit_emotion$subreddit=="depression"],
  y = reddit_emotion_loc[reddit_emotion$subreddit=="TodayIamHappy"]
)
test_CCR$p.value
```

```
[1] 1.981485e-153
```

```
test_CCR$estimate
```

```
mean of x mean of y
-5.543292 -2.492451
```

We find that posts in r/TodayIamHappy have more internal (as opposed to external) locus of control than those in r/depression ($p < .001$).

CCR: Key Takeaways

CCR is a powerful tool because it can apply existing questionnaires—a fundamental tool in psychological research—to modern automated text analysis. In this sense it makes effective use

of contextualized embeddings provided by state of the art LLMs. CCR thus allows psychologists to efficiently implement questionnaire-based research designs that nevertheless allow participants to compose responses in their own words. One limitation of CCR is its limited ability to generalize—it is thus contraindicated for texts that do not contain similar content and wording to the embedded questionnaire. Nevertheless the risk of questionnaire-ness described above can be mitigated by constructing an anchored vector, discussed below.

Correlational Anchored Vectors (CAV)

An anchored vector is simply a direction in the embedding space. Rather than finding this direction by subtracting a negative construct embedding from a positive one, as we have done so far in this guide, we can use machine learning to find the direction that best represents the construct in a training dataset. This provides an alternative to DDR and CCR, by which construct representations are learned directly from examples. Such examples may be the product of human raters or of an experimental manipulation. For the example below we use data from Kasprzyk and Calin-Jageman (2014), who asked participants to write about either incidents in which they had power over others (Power) or incidents in which other people had power over them (Control).

```
# Embed Training Data
d_train <- read_csv("example_data/power_narratives.csv") |>
  mutate(condition = factor(condition, levels = c("Control", "Power")))


d_train_bge <- textEmbed(
  d_train$text,
  model = "BAAI/bge-base-en-v1.5", # model name
  tokens_select = "[CLS]", # select only [CLS] token embedding
  layers = -1,  # last layer
  dim_name = FALSE,
  keep_token_embeddings = FALSE
```

```
)


# Rejoin Embeddings to Labelled Training Data
d_train <- d_train |>
  bind_cols(d_train_bge$texts[[1]])
```

With Partial Least Squares (PLS) regression (Mevik & Wehrens, 2007; Wold et al., 2001), which finds directions in the feature space that best correlate with the dependent variable (in this case Power as opposed to Control), we create a CAV. PLS is implemented here with the *caret* package (Kuhn & Max, 2008).

```
# Partial Least Squares (PLS) regression
set.seed(2024)
pls_control <- caret::train(
  condition ~ .,
  data = select(d_train, condition, Dim1:Dim768),
  method = "pls",
  scale = FALSE,  # keep original embedding dimensions
  trControl = caret::trainControl("cv", number = 10),  # cross-validation
  tuneLength = 1  # only 1 component (our anchored vector)
)
```

```
# Extract CAV
projection <- pls_control$finalModel$projection[,1]
power_loading <- pls_control$finalModel$Yloadings["Power",1]
power_cav <- projection*power_loading


# Calculate Distance Metrics
```

```
reddit_emotion_control <- reddit_emotion_bge |>
  rowwise() |>
  mutate(control = dot_prod(c_across(Dim1:Dim768), power_cav)) |>
  pull(control)


# Test Hypothesis
test_correlational <- t.test(
  x = reddit_emotion_control[reddit_emotion$subreddit=="depression"],
  y = reddit_emotion_control[reddit_emotion$subreddit=="TodayIamHappy"]
)
test_correlational$p.value
```

```
[1] 0
```

```
test_correlational$estimate
```

```
 mean of x  mean of y
-0.3093040  0.1046436
```

We find that posts in r/TodayIamHappy share more characteristics in common with narratives of the speaker having control over others than did posts in r/depression ($p < .001$).

CAV: Key Takeaways

CAV allows the application of closely related texts. Given a training set that closely resembles the texts of interest, CAV provides a relatively simple metric that is likely to be highly accurate and unconfounded by differences in text genre. On the other hand, CAV requires applicable training data and introduces questions of validity when applying patterns in the training set to the texts of interest.

Other Machine Learning Methods

Using PLS regression to generate a CAV is a very simple way to leverage machine learning for interpreting embeddings. The simplicity of dealing with a single direction in embedding space can provide useful intuition and transparency for researchers. Nevertheless, it sacrifices some of the predictive capacities available with modern machine learning methods. With more complex algorithms, the nonlinearity of contextualized embedding spaces becomes less of a problem. Given enough training data, one can specify a model that can capture nonlinear patterns, such as a support vector machine. More complex methods also allow the simultaneous use of embeddings from multiple layers of an LLM. A comprehensive review of machine learning algorithms for psychological features is beyond the scope of this guide. See Kjell et al. (2022) for a useful introductory primer.

Conclusion

In this guide, we have introduced the fundamentals of neural word embedding models including word2vec, GloVe, and fastText. In doing so, we have put special emphasis on the geometric properties of the embeddings created by these models. For example, the correspondence between the magnitude of a word embedding and its semantic informativeness is guaranteed by the architecture of such models. This correspondence also forms the basis for the much-touted additive analogical property of word embeddings (Ethayarajh et al., 2019; Mikolov, Chen, et al., 2013). This lies in contrast to embeddings generated by LLMs, which may have non-linear or non-zero-centered organizational patterns (Cai et al., 2021). Decontextualized word embedding models therefore have two major advantages when compared to LLM embeddings: their reliable geometries and their computational efficiency. On the other hand, the contextualizing abilities of LLMs can often make up for such deficiencies, especially with the help of recent models specialized for use in semantic embedding.

With the necessary background provided above, we have reviewed three methods of quantifying psychological constructs in embedding spaces: DDR, CCR, and CAV. We recommend DDR for abstract constructs relating to the overall feel of a text, especially when the

research requires that these constructs generalize to various genres of text. When using dictionaries validated for use in word counting, we recommend weighting words by their frequency when computing the average DDR. We recommend CCR for cases in which texts are relatively similar in content to the embedded questionnaire, such as experiments in which participants are asked to respond to a related prompt. CAV is appropriate only when suitably large and reliable training data is available.

References

Allport, G. W. (1942). The use of personal documents in psychological science. *Social Science Research Council Bulletin*, *49*, xix + 210–xix + 210.

Atari, M., Omrani, A., & Dehghani, M. (2023). *Contextualized construct representation: Leveraging psychometric scales to advance theory-driven text analysis*. PsyArXiv. https://doi.org/10.31234/osf.io/m93pd

Baayen, R. H. (2001). *Word frequency distributions*. Springer Netherlands. https://link.springer.com/book/10.1007/978-94-010-0844-0

Baumgartner, J., Zannettou, S., Keegan, B., Squire, M., & Blackburn, J. (2020). *The pushshift reddit dataset*. Zenodo. https://doi.org/10.5281/zenodo.3608135

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., & Matsuo, A. (2018). Quanteda: An r package for the quantitative analysis of textual data. *Journal of Open Source Software*, *3*(30), 774. https://doi.org/10.21105/joss.00774

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). *Enriching word vectors with subword information*. https://arxiv.org/abs/1607.04606

Boyd, R., Ashokkumar, A., Seraj, S., & Pennebaker, J. (2022). *The development and psychometric properties of LIWC-22*. https://doi.org/10.13140/RG.2.2.23890.43205

Cai, X., Huang, J., Bian, Y., & Church, K. (2021). Isotropy in the contextual embedding space: Clusters and manifolds. *International Conference on Learning Representations*. https://openreview.net/forum?id=xYGNO86OWDH

Creegan, R. F. (1944). The phenomenological analysis of personal documents. *The Journal of*

Abnormal and Social Psychology, *39*(2), 244–266. https://doi.org/10.1037/h0062816

Crystal, D. (1997). *The cambridge encyclopedia of language* (Second edition.). Cambridge University Press.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the Association for Information Science and Technology*, *41*(6), 391–407. https://doi.org/10.1002/(SICI)1097-4571(199009)41:6%3C391::AID-ASI1%3E3.0.CO;2-9

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding*. https://arxiv.org/abs/1810.04805

Ethayarajh, K. (2019). *How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings*. https://arxiv.org/abs/1909.00512

Ethayarajh, K., Duvenaud, D., & Hirst, G. (2019). *Towards understanding linear word analogies*. https://arxiv.org/abs/1810.04882

Frimer, J. A., Boghrati, R., Haidt, J., Graham, J., & Dehgani, M. (2019). Moral foundations dictionary for linguistic analyses 2.0. *Unpublished Manuscript*.

Gao, J., He, D., Tan, X., Qin, T., Wang, L., & Liu, T.-Y. (2019). *Representation degeneration problem in training natural language generation models*. https://arxiv.org/abs/1907.12009

Garten, J., Hoover, J., Johnson, K. M., Boghrati, R., Iskiwitch, C., & Dehghani, M. (2018). Dictionaries and distributions: Combining expert knowledge and large scale textual data content analysis: Distributed dictionary representation. *Behavior Research Methods*, *50*, 344–361.

Goldberg, Y., & Levy, O. (2014). *word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method*. https://arxiv.org/abs/1402.3722

Grand, G., Blank, I. A., Pereira, F., & Fedorenko, E. (2022). Semantic projection recovers rich human knowledge of multiple object features from word embeddings. *Nature Human Behaviour*, *6*(7), 975–987. https://doi.org/10.1038/s41562-022-01316-8

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., & Mikolov, T. (2018). Learning word vectors for

157 languages. *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. https://fasttext.cc/docs/en/crawl-vectors.html

Harris, Z. S. (1954). Distributional Structure. *WORD*, *10*(2-3), 146–162. https://doi.org/10.1080/00437956.1954.11659520

Kasprzyk, L., & Calin-Jageman, R. (2014). The effect of power on visual perspective: Replications of galinksy, et al. 2006. *Preprint at Https://Doi. Org/10.17605/OSF. IO/WCH5R*.

Kjell, O., Giorgi, S., & Schwartz, H. A. (2021). *The text-package: An r-package for analyzing and visualizing human language using natural language processing and deep learning*. PsyArXiv. https://doi.org/10.31234/osf.io/293kt

Kjell, O., Sikström, S., Kjell, K., & Schwartz, H. (2022). Natural language analyzed with AI-based transformers predict traditional subjective well-being measures approaching the theoretical upper limits in accuracy. *Scientific Reports*, *12*, 3918. https://doi.org/10.1038/s41598-022-07520-w

Kuhn, & Max. (2008). Building predictive models in r using the caret package. *Journal of Statistical Software*, *28*(5), 1–26. https://doi.org/10.18637/jss.v028.i05

Li, B., Zhou, H., He, J., Wang, M., Yang, Y., & Li, L. (2020). *On the sentence embeddings from pre-trained language models*. https://arxiv.org/abs/2011.05864

Mevik, B.-H., & Wehrens, R. (2007). The pls package: Principal component and partial least squares regression in r. *Journal of Statistical Software*, *18*(2), 1–23. https://doi.org/10.18637/jss.v018.i02

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. https://arxiv.org/abs/1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). *Distributed representations of words and phrases and their compositionality*. https://arxiv.org/abs/1310.4546

Mikolov, T., Yih, W., & Zweig, G. (2013). Linguistic regularities in continuous space word representations. In L. Vanderwende, H. Daumé III, & K. Kirchhoff (Eds.), *Proceedings of the 2013 conference of the north American chapter of the association for computational*

linguistics: Human language technologies (pp. 746–751). Association for Computational Linguistics. https://aclanthology.org/N13-1090

Muennighoff, N., Tazi, N., Magne, L., & Reimers, N. (2022). MTEB: Massive text embedding benchmark. *arXiv Preprint arXiv:2210.07316*. https://doi.org/10.48550/ARXIV.2210.07316

O'Connell, D., & Kowal, S. (2003). Psycholinguistics: A half century of monologism. *The American Journal of Psychology*, *116*, 191–212. https://doi.org/10.2307/1423577

O'Connor, B. (2012). Cosine similarity, Pearson correlation, and OLS coefficients. In *AI and Social Science*. https://brenocon.com/blog/2012/03/cosine-similarity-pearson-correlation-and-ols-coefficients/

Oyama, M., Yokoi, S., & Shimodaira, H. (2023). *Norm of word embedding encodes information gain*. https://arxiv.org/abs/2212.09663

Pang, B., & Lee, L. (2005). *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales*. https://arxiv.org/abs/cs/0506075

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. http://www.aclweb.org/anthology/D14-1162

Peterson, C., & Seligman, M. E. (1984). Causal explanations as a risk factor for depression: Theory and evidence. *Psychological Review*, *91*(3), 347–374.

Placiński, M., & Żywiczyński, P. (2023). Modality effect in interactive alignment: Differences between spoken and text-based conversation. *Lingua*, *293*, 103592. https://doi.org/https://doi.org/10.1016/j.lingua.2023.103592

Rana, A. (2010). *Common crawl – building an open web-scale crawl using hadoop*. https://www.slideshare.net/hadoopusergroup/common-crawlpresentation

Reif, E., Yuan, A., Wattenberg, M., Viegas, F. B., Coenen, A., Pearce, A., & Kim, B. (2019). Visualizing and measuring the geometry of BERT. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc. https://proceedings.neurips.cc/paper_

files/paper/2019/file/159c1ffe5b61b41b3c4d8f4c2150f6c4-Paper.pdf

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)* (pp. 3982–3992). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

Rotter, J. B. (1966). Generalized expectancies for internal versus external control of reinforcement. *Psychological Monographs: General and Applied*, *80*(1), 1.

Rude, S., Gortner, E.-M., & Pennebaker, J. (2004). Language use of depressed and depression-vulnerable college students. *Cognition & Emotion - COGNITION EMOTION*, *18*, 1121–1133. https://doi.org/10.1080/02699930441000030

Schakel, A. M. J., & Wilson, B. J. (2015). *Measuring word significance using distributed representations of words*. https://arxiv.org/abs/1508.02297

Simchon, A., Hadar, B., & Gilead, M. (2023). A computational text analysis investigation of the relation between personal and linguistic agency. *Communications Psychology*, 1–9. https://doi.org/10.1038/s44271-023-00020-1

Teitelbaum, L., & Simchon, A. (2024). *Data science for psychology: Natural language*. https://doi.org/https://zenodo.org/doi/10.5281/zenodo.10908366

Torabi Asr, F., Zinkov, R., & Jones, M. (2018). Querying word embeddings for similarity and relatedness. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)* (pp. 675–684). Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1062

Trager, J., Ziabari, A. S., Davani, A. M., Golazazian, P., Karimi-Malekabadi, F., Omrani, A., Li, Z., Kennedy, B., Reimer, N. K., Reyes, M., Cheng, K., Wei, M., Merrifield, C., Khosravi, A., Alvarez, E., & Dehghani, M. (2022). *The moral foundations reddit corpus*. https://arxiv.org/abs/2208.05545

Tversky, A., & Gati, I. (1982). Similarity, separability, and the triangle inequality. *Psychological Review*, *89*, 123–154. https://doi.org/10.1037/0033-295X.89.2.123

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention is all you need*. https://arxiv.org/abs/1706.03762

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Wold, S., Sjöström, M., & Eriksson, L. (2001). PLS-regression: A basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, *58*, 109–130. https://api.semanticscholar.org/CorpusID:11920190

Xiao, S., Liu, Z., Zhang, P., & Muennighoff, N. (2023). *C-pack: Packaged resources to advance general chinese embedding*. https://arxiv.org/abs/2309.07597

Appendix

DDR Metrics

Garten et al. (2018) found that DDR works best with smaller dictionaries of only the words most directly connected to the construct being measured (around 30 words worked best in their experiments). Here we replicate their study using slightly different methods, and extend it to a variety of vector-based metrics, including anchored vectors. We also investigate the impact of weighting the averaged dictionary representation by token frequency, which we suggested would eliminate the observed superiority of smaller dictionaries.

Benchmark 1: Negative Sentiment in Movie Reviews

As an initial benchmark, we used the same data used by Garten et al. (2018) in their investigation of dictionary size: a set of 2000 movie reviews, half labeled as negative and half as positive (Pang & Lee, 2005). For vector representations of words and texts, we used a publicly available GloVe model trained on 2B Tweets to produce 100-dimensional embeddings (Pennington et al., 2014). For construct representations, we used the positive tone and negative tone dictionaries from LIWC-22 (Boyd et al., 2022), expanded on the movie reviews dataset. The primary DDR was the average embedding of the negative tone dictionary, while for anchored vectors the average embedding of the positive tone dictionary was used as a second anchor.

We investigated the following metrics:

- **Cosine similarity** with the negative tone DDR
- **Cosine similarity with the anchored vector** (equivalent to projection of the normalized text vector onto the anchored vector)
- **Dot product** with the negative tone DDR
- **Dot product with the anchored vector** (equivalent to projection of the raw text vector onto the anchored vector)
- **Dot product with the pre-normalized anchored vector** (i.e. positive and negative DDR embeddings normalized before calculating the anchored vector)
- **Euclidean distance** from the negative tone DDR

- **Euclidean distance from the anchored vector**

To evaluate the predictive value of each metric, we trained a univariate logistic regression model for each metric at each dictionary size. We then computed an F1 score for the model's predictions on the training set, with the model's threshold set at 0.5 (i.e. any text given a probability of greater than 0.5 of being negative was considered as having been predicted to be negative).

When using only the primary DDR, we found that the performance of equal weighting drops sharply with increasing dictionary size, while the performance of frequency weighting continues to rise. Indeed, DDRs computed with equal weighting were likely to result in negative associations between the predictor and the outcome, even at small dictionary sizes. Surprisingly, metrics based on anchored vectors were robust to this instability. Additionally, frequency weighting was not superior to equal weighting for metrics based on anchored vectors. The overall best performing metric across all dictionary sizes was cosine similarity with the anchored vector calculated using equal weighting.

The pattern of results was similar among the highest performing dictionaries at each size, with two notable exceptions. First, frequency weighting was slightly better than equal weighting for all metrics (including those based on anchored vectors) except cosine similarity with the anchored vector. Second, the performance of frequency weighting among metrics using only the primary DDR did not increase with sample size.

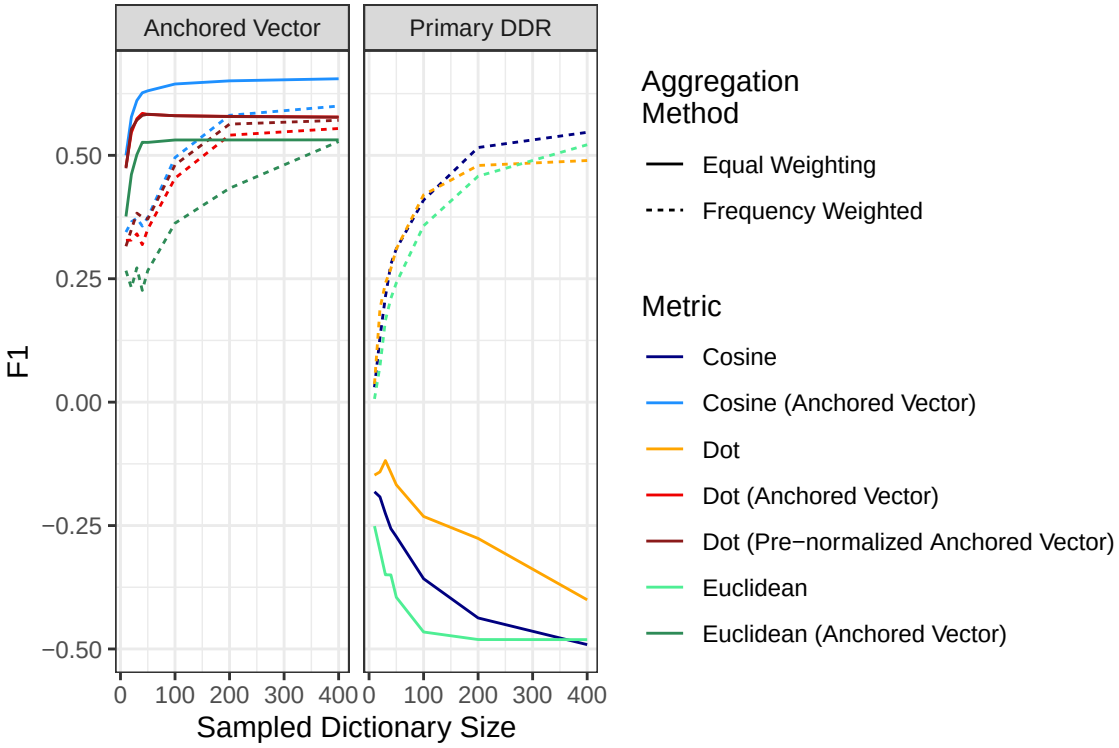
As a further validation of our proposed frequency weighting method, we investigated the performance of dictionary representations as a function of the variance of their word frequencies. We found that dictionaries with higher variation in word frequencies result in a large advantage for frequency weighted aggregation. As before, we found that this did not hold for metrics based on anchored vectors.

Benchmark 2: Moral Foundations in Reddit Comments

To see whether the patterns observed for negative valence extend to more complex psychological constructs, we evaluated the same metrics and aggregation methods on a large

Figure A1

Mean F1 by Dictionary Size



Note. Each data point represents the mean of 200 samples. F1 scores arising from negative associations between the predictor and the outcome are counted as negative.

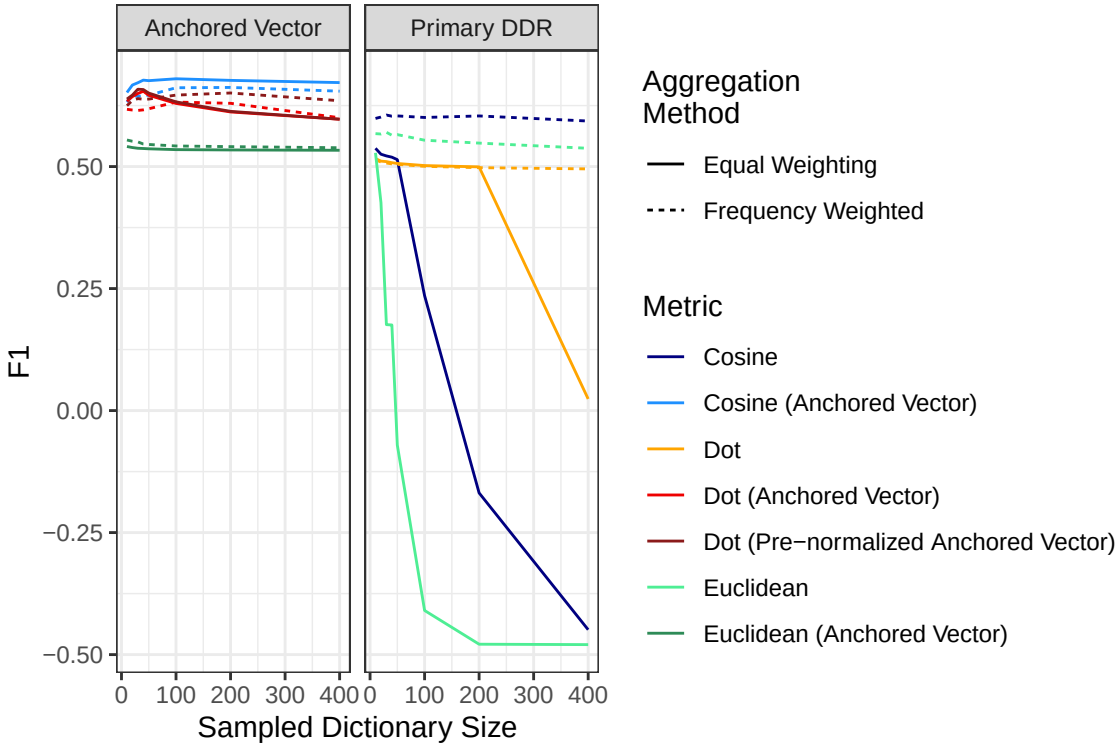
dataset of Reddit comments (N = 16,123), which were manually annotated with six moral foundations: authority, care, fairness, loyalty, sanctity, and vice (Trager et al., 2022). Each text was annotated by at least three annotators, giving a total of 53,545 cases.

To construct distributed construct representations for the moral foundations, we used the Moral Foundations Dictionary 2.0 (Frimer et al., 2019), which includes more than 200 words per foundation. Since the moral foundations do not have clear opposites, we constructed a neutral embedding by averaging the embeddings of all six foundations. This neutral embedding was used as the second anchor in anchored vector metrics.

Since the Reddit dataset was heavily imbalanced (i.e. most comments were not labeled as reflecting any given foundation), we set the classifier threshold at the empirical probability of each

Figure A2

Mean F1 Score by Dictionary Size for Dictionaries Above the 80th Percentile



Note. Each data point represents the mean of 200 samples. F1 scores arising from negative associations between the predictor and the outcome are counted as negative.

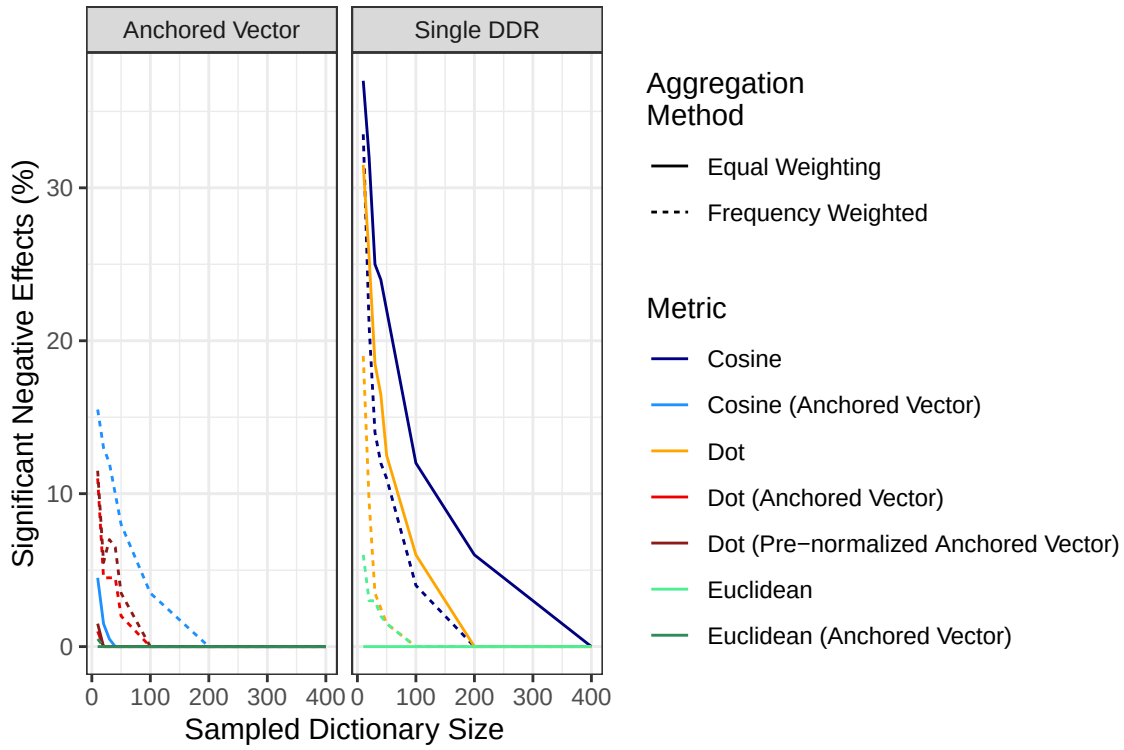
rating. For example, if 20% of texts in the dataset were labeled as reflecting loyalty, we would consider any text given a probability of greater than 0.2 to reflect loyalty according to the model. Using these predictions, F1 scores were computed as for the first benchmark.

We found that frequency weighted aggregation performed better than equal weighting in all five moral foundations except loyalty, for which all metrics performed comparably. Furthermore, the pattern of optimal metrics was somewhat erratic for equal weighting, whereas with frequency weighted aggregation, cosine similarity with the DDR performed best in all five foundations.

Curiously, anchored vector metrics did not perform as well as simple similarity scores. This may be attributable to the use of a neutral embedding as the second anchor, rather than a true

Figure A3

Significant Negative Effects by Dictionary Size



Note. 200 samples per data point.

opposite.

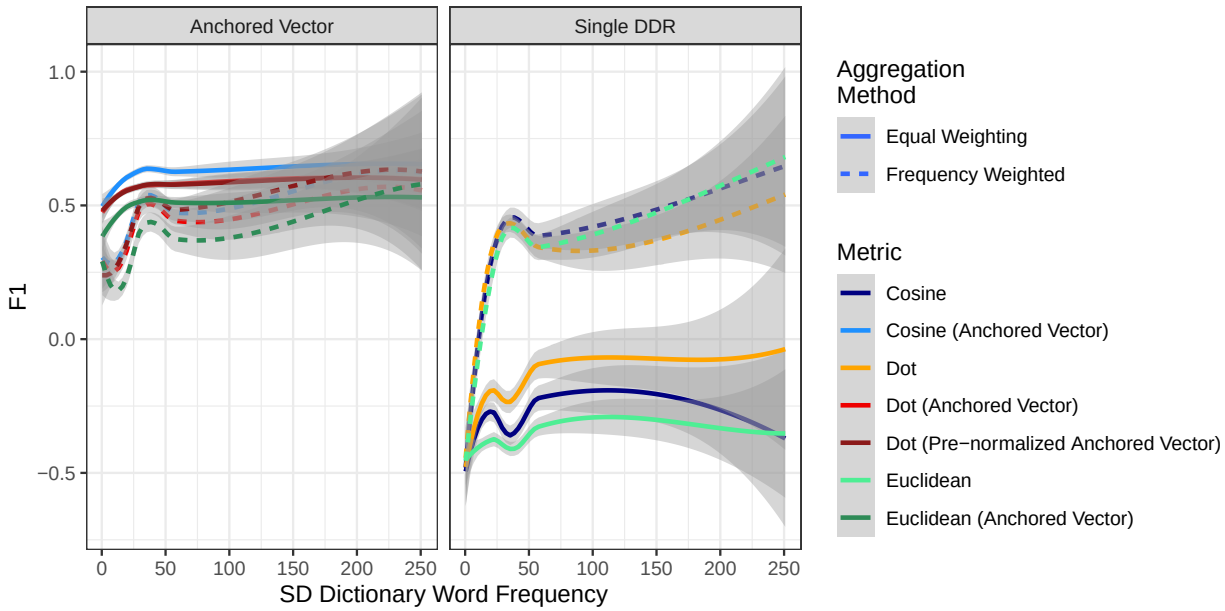
For equal weighting, Euclidean distance from the anchored vector resulted in significant negative effects in 1/5 foundations. For frequency weighted aggregation, the dot product with the pre-normalized anchored vector resulted in significant negative effects in 3/5 foundations. Otherwise no negative effects were observed.

Conclusions

The results of our experiments support our suggestion that the diminishing performance of larger dictionaries is due to the influence of less frequent words. We found that computing the DDR as a weighted average generally improves performance, especially for larger dictionaries of a few hundred words. We further found that metrics based on anchored vectors were largely robust to the influence of term weighting. Nevertheless, metrics based on anchored vectors require a

Figure A4

Mean F1 Score by Dictionary Frequency Variance

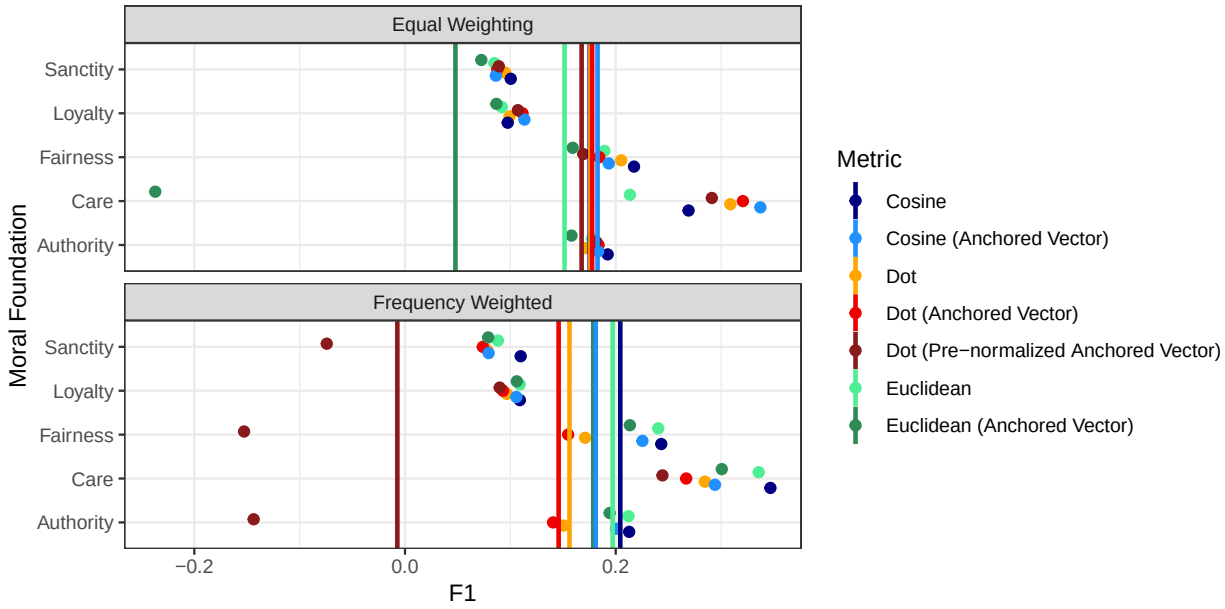


Note. Smoothing lines are computed with LOESS regression. F1 scores arising from negative associations between the predictor and the outcome are counted as negative.

construct with a clear opposite—while anchored vectors performed well for negative vs. positive sentiment, they did not perform well for moral foundations as compared to a neutral moral foundation embedding.

Figure A5

Equal vs. Frequency Weights for Moral Foundation DDRs



Note. Vertical lines represent mean F1 scores across moral foundations.

Appendix

CCR Metrics

Atari et al. (2023) asked participants to describe their core values in their own words (values essay) and to likewise describe their activities in the past week (behaviors essay). They then assessed participants on 22 questionnaire-based scales. This design allowed them to validate CCR by obtaining a contextualized embedding of each questionnaire and comparing it to the contextualized embedding of each essay. In benchmark 1 we partially replicate Atari et al.'s analysis and extend it to various vector-based metrics, including anchored vectors. In benchmark 2, we investigate the extent to which the techniques generalize to a more naturalistic context.

Benchmark 1: Prompted Essays

The anchored vector is equivalent to the embedding of the questionnaire (positive CCR) minus the embedding of the negated questionnaire (negative CCR). The individualism items were used as a negated form of the collectivism, and vice versa. To obtain negated versions of the remaining questionnaires, we queried GPT-4o and manually curated the results.[25] We investigated the following metrics:

- **Cosine similarity** with the CCR
- **Cosine similarity with the anchored vector** (equivalent to projection of the normalized text vector onto the anchored vector)
- **Dot product** with the CCR
- **Dot product with the anchored vector** (equivalent to projection of the raw text vector onto the anchored vector)
- **Dot product with the pre-normalized anchored vector** (i.e. positive and negative CCR embeddings normalized before calculating the anchored vector)
- **Euclidean distance** from the CCR
- **Euclidean distance from the anchored vector**

[25] All matrials, including original and negated questionnaire items, are available on our Github repo at https://github.com/rimonim/embeddings_tutorial.

Although the term CCR implies the use of contextualized embeddings, we use it here to refer to any vector embedding of a questionnaire. To obtain embeddings for participant essays and questionnaire items, we used two pretrained models:

- **SBERT** (`sentence-transformers/all-MiniLM-L12-v2`), a model similar to that used by Atari et al. (2023), designed to produce 384-dimensional contextualized embeddings amenable to use with cosine similarity.
- **GloVe** (`glove.twitter.27B.100d`) a pretrained word embedding model with a 100-dimensional embedding space. GloVe embeddings each text were obtained by aggregating the embeddings of each word in the text, discounting tokens not available in the pretrained GloVe model. Since GloVe is a decontextualized model, we used it here as a baseline for evaluating the utility of contextualized embeddings in CCR analysis.

As a further baseline, we also performed DDR using dictionaries for each construct and its opposite. These dictionaries were generated by GPT-4o and manually curated, but were not validated in any way prior to testing. We can therefore consider this DDR to be a conservative baseline for evaluating CCR methods. DDR analysis was performed with the same GloVe model described above.

All code is available in the source code for this appendix on Github.

Results: Values Essay

In predicting questionnaire responses using participant values essays, the most effective metrics were as follows:

DDR:

- Dot product with anchored vector (mean $R^2 = 0.016$)
- Dot product with pre-normalized anchored vector (mean $R^2 = 0.015$)
- Cosine similarity with anchored vector (mean $R^2 = 0.012$)

GloVe:

- Cosine similarity with anchored vector (mean $R^2 = 0.010$)
- Dot product with pre-normalized anchored vector (mean $R^2 = 0.009$)

SBERT:

- Dot product with anchored vector (mean $R^2 = 0.023$)
- Cosine similarity with anchored vector (mean $R^2 = 0.023$)
- Dot product with pre-normalized anchored vector (mean $R^2 = 0.022$)

For values essays, SBERT was consistently more effective than GloVe, and generally better than DDR. In all models, Euclidean distance metrics were minimally effective and most likely to result in significant negative effects. The success of GloVe-based CCR with metrics that involve anchored vectors is surprising, since most negated questionnaire items only differ from the originals by the words "do not" or similarly uninformative negations.

Results: Behaviors Essay

In predicting questionnaire responses using participant behaviors essays, the most effective metrics were as follows:

DDR:

- Dot product with anchored vector (mean $R^2 = 0.017$)
- Dot product with CCR (mean $R^2 = 0.016$)
- Euclidean distance from anchored vector (mean $R^2 = 0.014$)

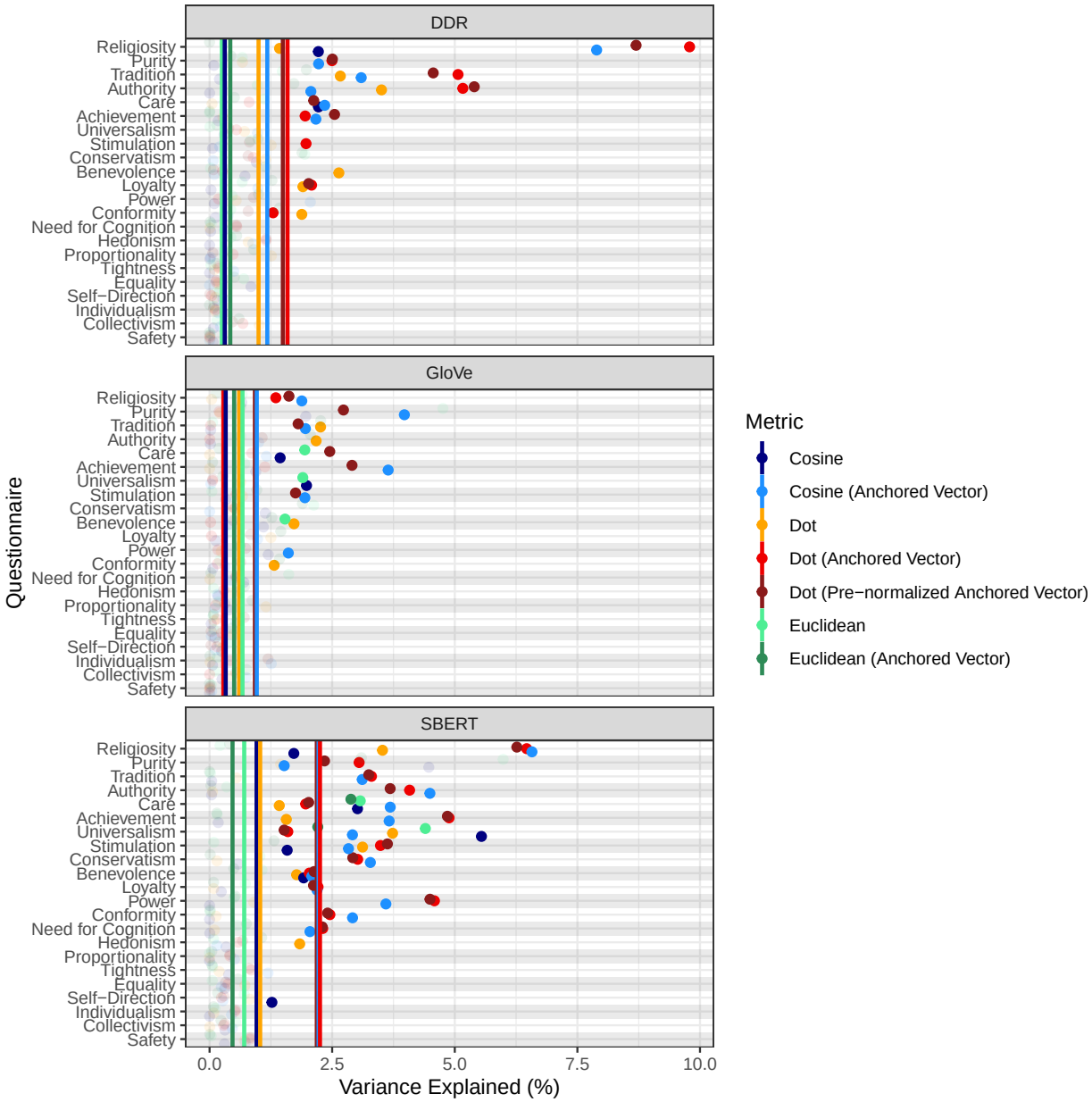
GloVe:

- Dot product with CCR (mean $R^2 = 0.014$)
- Euclidean distance from anchored vector (mean $R^2 = 0.014$)
- Dot product with anchored vector (mean $R^2 = 0.006$)

SBERT:

Figure B1

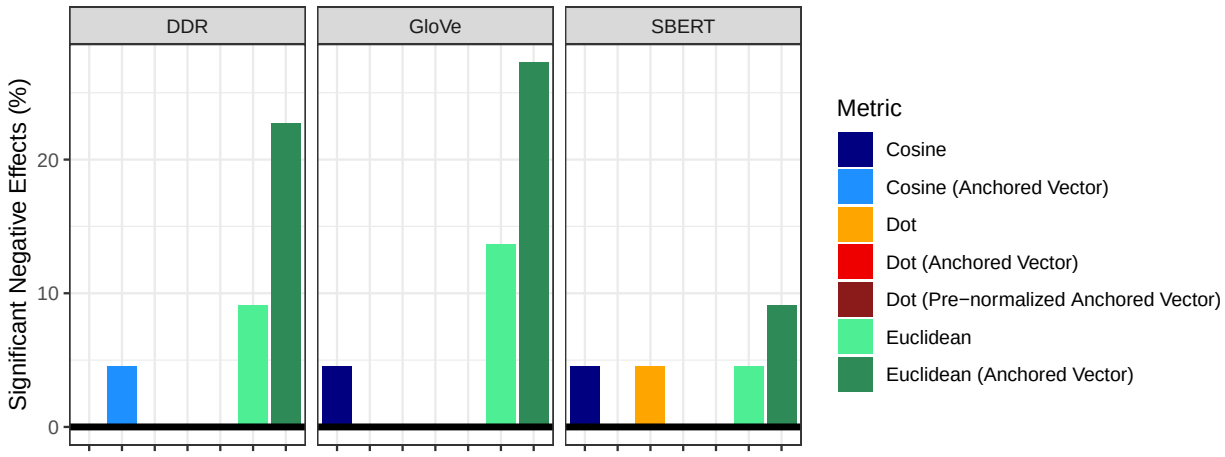
Values Essay



Note. Negative or insignificant effects ($p > 0.05$) are displayed as translucent and are considered to be 0 in metric-wise averages.

Figure B2

Significant Negative Effects in Values Essay



- Euclidean distance from anchored vector (mean $R^2 = 0.008$)
- Dot product with CCR (mean $R^2 = 0.006$)

For behaviors essays, GloVe-based CCR was more consistently effective than SBERT, and DDR was most effective overall. While Euclidean distance from the anchored vector was most effective on average for both models, it was also most likely to result in significant negative effects.

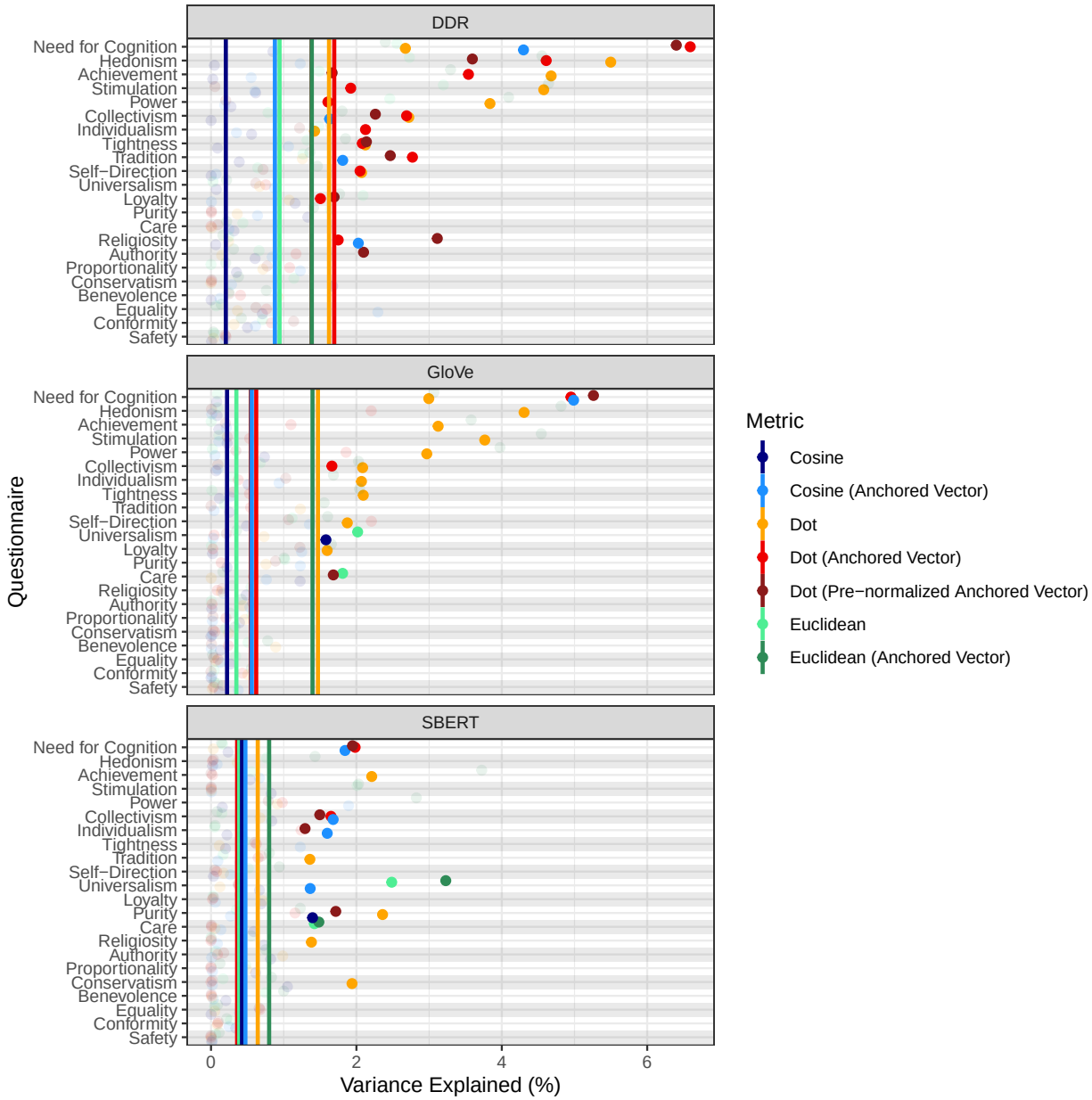
Conclusions

CCR (Atari et al., 2023) proposes to measure psychological constructs in texts by comparing text embeddings to embeddings of questionnaires. This approach to text analysis can be highly effective, but is sensitive to the nature of both questionnaires and texts. Some questionnaires do not appear to be amenable to CCR at all, including those used here for tightness, collectivism, individualism, proportionality, equality, and safety. Among those scales that were amenable to CCR metrics, the pattern of optimal metrics was greatly influenced by the content of the texts being analyzed.

Values essays tend to be similar in content to the values questionnaires being used, since the questionnaire items almost all consist of statements in the first person. Contextualized embeddings from an SBERT model appear to perform best in these sorts of situations.

Figure B3

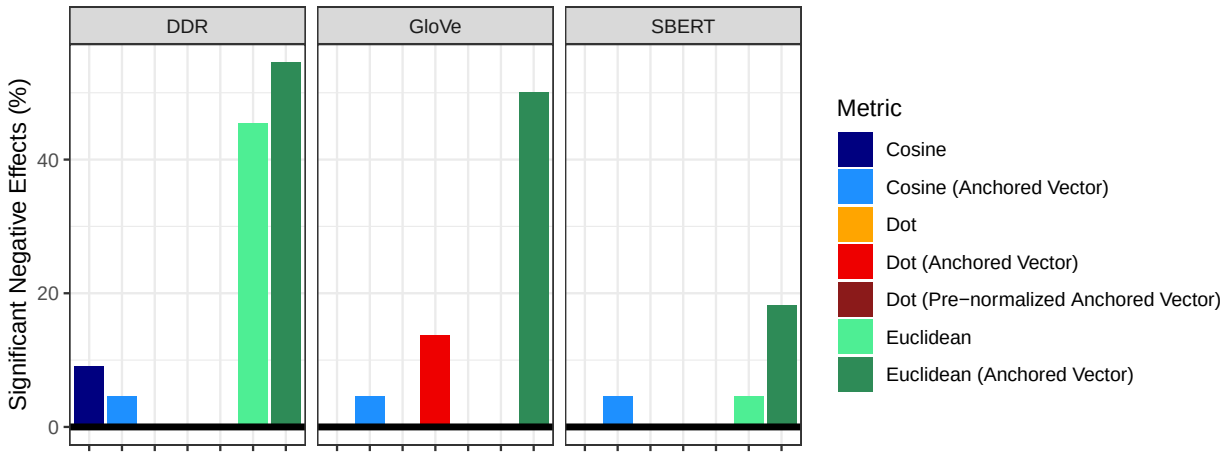
Behaviors Essay



Note. Negative or insignificant effects ($p > 0.05$) are displayed as translucent and are considered to be 0 in metric-wise averages.

Figure B4

Significant Negative Effects in Behaviors Essay



Behaviors essays, which bear little resemblance to questionnaire items in either tone or content, behave very differently. In particular, GloVe embeddings of the questionnaires tend to perform better. This may be due to the consistent geometric properties of GloVe embeddings, which can more reliably encode semantic relationships between very different contexts. Alternatively, this may be due to the datasets used to train SBERT models, which emphasize topical similarity rather than similarity in tone (Reimers & Gurevych, 2019). Unsurprisingly, GloVe embeddings of dictionaries associated with the questionnaires performed best overall.

Overall, the results of this analysis suggest that CCR is can be a powerful tool, but that it should be used primarily in contexts in which the content of the questionnaires is similar to that of the texts being analyzed. In such cases, we recommend negating the questionnaire items, computing an anchored vector, and scoring texts by the dot products of their embeddings with the anchored vector.

When the content of the questionnaires is not similar to that of the texts being analyzed, we recommend using DDR with negative and positive versions of each dictionary, and scoring texts by the dot products of their embeddings with the anchored vector. This approach appears to outperform CCR even with unvalidated dictionaries generated by GPT-4o.