9.19: Computational Psycholinguistics, Pset 5
due 5 November 2021

22 October 2021

1 Logistic Regression

The Colab notebook for this problem is available at https://colab.research.google.com/drive/19IGCEGjgg9i2Z6EQhj6-PGrpaQvREtcV?usp=sharing.

This problem is a continuation of our look at the English dative alternation. You will work with a dataset of annotated examples of the dative alternation from spontaneous speech in conversation, reported in Bresnan et al. (2007) and made available through the `languageR` package (Baayen, 2007). The Colab notebook for this problem automatically downloads this dataset as `dative.csv`.

As you've seen before, the dative alternation involves DITRANSITIVE verbs, and

(1) a. Kim mailed $\overbrace{\text{me}}^{\textsc{Recipient}}$ $\overbrace{\text{a gift}}^{\textsc{Theme}}$. [D(OUBLE) O(BJECT)]
b. Kim mailed $\underbrace{\text{a gift}}_{\textsc{Theme}}$ to $\underbrace{\text{me}}_{\textsc{Recipient}}$. [P(REPOSITIONAL) D(ATIVE)]

The THEME and RECIPIENT arguments are, respectively, what gets acted upon (usually transferred in physical location or possession), and the recipient or destination of the action. One qualitative intuition often reported about the dative alternation is that cases where the recipient argument is a large phrase (as measured, e.g., in number of words) are awkward, such as the below example:

(2) ?Kim mailed everyone who had attended the party yesterday a gift.

Alternatively, some researchers have proposed that what is more crucial is whether the recipient and theme arguments are pronouns. In this problem, we test these ideas by building simple logistic regression models of the dative alternation to look at the predictive effects of the length and pronominality of the recipient and theme arguments. We will use the `pandas` and `statmodels` Python packages for this.

In general, in studying the factors influencing speaker preference in the dative alternation, we will be interested in estimating the following probabilistic model:

$$P(\text{Construction} = \text{Double Object}|\text{Subject}, \text{Verb}, \text{Recipient}, \text{Theme})$$

where any of a number of features of the subject, verb, recipient, and theme might influence the speaker or writer's choice of linguistic construction. For purposes of this problem, however, we will dramatically simplify. First, we will ignore the subject and verb together, simplifying our problem to:[1]

$$P(\text{Construction} = \text{Double Object}|\text{Recipient}, \text{Theme})$$

Also, we'll only use a few features of the recipient and theme in our predictive model. Recall that logistic regression is characterized by the following equations:

$$\eta = \sum_i \beta_i X_i \qquad \text{(linear predictor)}$$

$$P(\text{success}) = \frac{e^\eta}{1 + e^\eta} \qquad \text{(logistic transform of linear predictor)}$$

For the dative alternation, the two possible outcomes are the double object construction (**DO**) and prepositional dative construction (**PD**). We arbitrarily choose DO as the outcome corresponding to"success".

Unlike the case of binomial ordering choice, there is a systematic difference between the possible outcomes that holds across all instances of the dative alternation: it is always the same two constructions that are being chosen between. To capture the possibility of an overall preference for one construction or the other, we add what is called an "intercept" or "bias" term to the equation determining the linear predictor. This is often expressed in the statistics literature (assuming M predictors):

$$\eta = \alpha + \sum_{i=1}^{M} \beta_i X_i \qquad \text{(linear predictor)},$$

but it can equivalently be expressed by defining a "dummy" predictor X_0 whose value is always 1, and writing the linear predictor as:

$$\eta = \sum_{i=0}^{M} \beta_i X_i \qquad \text{(linear predictor)},$$

[1]This is an oversimplification, actually: in particular, the identity of the verb has a *lot* of predictive information. This information is most effectively brought into our model by introducing a hierarchical component like the one we studied in the last part of our in-class coverage of binoimals, endowing verbs with idiosyncratic preferences for which construction they prefer and by how much. This problem doesn't cover these models, but Bresnan et al. (2007) and Morgan and Levy (2015) are good references for seeing how to include this hierarchical (sometimes called "mixed-effects") component for the dative alternation and for binomials, respectively.

a formulation more common in the machine learning literature. (Note that it would be inappropriate to include an intercept in the model for binomial ordering preferences, because there is no intrinsic difference between "success" and "failure" outcomes that is consistently defined across different specific cases of binomial ordering choice.)

We can evaluate the quality of a logistic regression model in a couple of ways. One is predicting the CLASS of the outcome: here, DO or PD? We say that a logistic regression model predicts "success" for a datum if it assigns $P(\text{success}) > 0.5$ for that datum, otherwise "failure". A second is the LOG-LIKELIHOOD of the dataset—the summed log-probabilities of all the observations in the dataset under the fitted model.

Tasks:

1. Define and implement an 80/20 train/test random split of the `dative` dataset.

2. Fit a logistic regression model to the training set that uses *only* recipient pronominality and an intercept term. What is its classification accuracy on the held-out test dataset? How about its log-likelihood?

3. Add theme pronominality as a predictor to the model and see whether that improves the model's predictive power as assessed by held-out classification accuracy and log-likelihood.

4. Determine whether additionally adding theme and recipient length (in number of words) to the model further improves fit. Try both raw length or log-transformed length. Which gives better performance?

5. Look at the β coefficients of a fitted model with all four predictors and interpret them theoretically (don't worry about interpreting the intercept α, whose value will depend on the numeric coding scheme used for the predictors). Are there any general linguistic principles manifested in the values of all four predictors? Do you see any ways to simplify the model (reduce the number of predictor weights that have to be learned) based on general linguistic principles, without sacrificing much predictive accuracy? This may involve creating a new set of predictors that are a function of the four predictors you've been working with up until now.

2 Distance, similarity, and analogies in word embeddings

This problem set involves manipulating and using **word embeddings**: representations of the semantics of words as high-dimensional vectors. We will be using off-the-shelf semantic vectors derived in previous work (Pennington et al., 2014), called GloVe vectors. You should download one of the data zip files from https://nlp.stanford.edu/projects/glove/. We recommend `glove.6B.zip`, but you may use any of the vector files provided on the website. Note that working with larger vector files will make processing times in your code slower.

Unzipping the file `glove.6B.zip`, you will see a number of `.txt` files. For the exercises below, we recommend using the 300-dimensional vectors in the file `glove.6B.300d.txt`. (Note that all words the GloVe word vector files are converted to lower-case, so you will have to do the same in this exercise.)

The Colab notebook contains Python code for downloaing the GloVe vectors and reading them into a dictionary data structure. We call the resulting dictionary `e` (for embedding), so calling `e['car']` returns an array representing the semantics of the word *car*, and so on.

1. One of the main functions of semantic vectors is to represent similarity relations among words. For example, *frog* and *toad* are very similar in meaning, while *frog* and *yesterday* are very dissimilar.

   Write a function to compute the **cosine similarity** between two vectors. Cosine similarity is a score between -1 and 1 indicating similarity, where 1 is maximal similarity and -1 is minimal similarity. Cosine similarity between two vectors $\mathbf{A}$ and $\mathbf{B}$ is defined as:

   $$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}.$$

   Hint: You will probably find it faster and more convenient to use the function `numpy.dot` from the `numpy` package rather than manually implementing all the summations above!

   Solution: The file `analogies_solution.py` contains our solution to this problem. We implemented our cosine similarity function as:

   ```
   print(cosine_similarity(vectors["car"],vectors["truck"]))
   ```

   (a) Verify that your implementation of cosine similarity is correct by checking that it is **symmetrical**: it should be the case that `similarity(x,y) == similarity(y,x)` for all `x` and `y`. Demonstrate that this is the case with a few examples.

   Solution:

   ```
   print(cosine_similarity(vectors["truck"],vectors["car"]))
   ```

   ?? PythonTeX ??

   ```
   print(cosine_similarity(vectors["friend"],vectors["enemy"]))
   ```

   ?? PythonTeX ??

   ```
   print(cosine_similarity(vectors["enemy"],vectors["friend"]))
   ```

   ?? PythonTeX ??

   ```
   print(cosine_similarity(vectors["cow"],vectors["justice"]))
   ```

   ?? PythonTeX ??

```
print(cosine_similarity(vectors["justice"],vectors["cow"]))
```

?? PythonTeX ??

```
print(cosine_similarity(vectors["car"],vectors["truck"]),
      cosine_similarity(vectors["car"],vectors["person"]))
```

?? PythonTeX ??

(b) As sanity checks, verify that the following similarity relations are true in the GloVe vectors given your implementation of cosine similarity. Report the similarity relations for these examples.

i. *car* is closer to *truck* than to *person*
Solution:

```
print(cosine_similarity(vectors["mars"],vectors["venus"]),
      cosine_similarity(vectors["mars"],vectors["goes"]))
```

?? PythonTeX ??

ii. *Mars* is closer to *Venus* than to *goes*
Solution:

```
print(cosine_similarity(vectors["warm"],vectors["cool"]),
      cosine_similarity(vectors["warm"],vectors["yesterday"]))
```

?? PythonTeX ??

iii. *warm* is closer to *cool* than to *yesterday*
Solution:

```
print(cosine_similarity(vectors["red"],vectors["blue"]),
      cosine_similarity(vectors["red"],vectors["fast"]))
```

?? PythonTeX ??

iv. *red* is closer to *blue* than to *fast*
Solution:

```
print(cosine_similarity(vectors["pumpkin"],vectors["halloween"]),
      cosine_similarity(vectors["pumpkin"],vectors["christmas"]))
```

?? PythonTeX ??

v. Come up with two more examples that demonstrate correct similarity relations.
Solution:

```
print(cosine_similarity(vectors["edible"],vectors["fruit"]),
      cosine_similarity(vectors["edible"],vectors["rock"]))
```

?? PythonTeX ??

```
print(cosine_similarity(vectors["angel"],vectors["devil"]),
      cosine_similarity(vectors["angel"],vectors["god"]))
```

?? PythonTeX ??

vi. Come up with two examples where cosine similarity in the semantic vectors does not align with your intuitions about word similarity.

Solution:

```
print(cosine_similarity(vectors["hot"],vectors["scalding"]),
      cosine_similarity(vectors["hot"],vectors["cold"]))
```

?? PythonTeX ??

```
def analogy_vector(w1, w2, w3, vectors):
    return vectors[w2] - vectors[w1] + vectors[w3]

def analogy(w1, w2, w3, vectors, k=5):
    vector = analogy_vector(w1, w2, w3, vectors)
    distances = [(-cosine_similarity(vectors[w], vector), w) for
        w in vectors.keys()]
    return [w for _, w in sorted(distances)[:k]]
```

?? PythonTeX ??

(c) For the examples where cosine similarity does not match your intuitions, what do you think went wrong?

Solution:

For the first example, I think that "angel" and "god" are more semantically similar to one another in the most salient semantic dimension for me, namely goodness/badness. But the expression "angel and devil" (and its reverse, "devil and angel"), is higher-frequency than any collocation involving "angel" and "god", so the model tries hard to make "angel" and "devil" to have similar word embeddings so that the words are predicted to co-occur near each other.

For the second example, I suspect the problem is as follows. The word "scalding" is a more constrained version of "hot", both in temperature range and in terms of the kinds of experiences one could have with the predicated object (for something to be "scalding" it has to be possible to contact your skin and scald you). The word "cold" in contrast means the *opposite* of "hot", but it has a very similar distribution to "hot" because it is not as constrained as "scalding" (plus the two are used in collocation as in "hot or cold").

(d) **Extra credit**: Try a different distance metric, such as Euclidean distance. Does it result in qualitatively different patterns on your test suite?

2. Write a function to perform the **analogy task**: Given words w_1, w_2, and w_3, find a word x such that $w_1 : w_2 :: w_3 : x$. For example, for the analogy problem *France*:*Paris* :: *England*:x, the answer should be *London*. To solve analogies using semantic vectors, letting $e(w)$ indicate the embedding for a word w, calculate a vector $y = e(w_2) - e(w_1) + e(w_3)$ and find the word whose vector is closest to y.

(a) Explain why the analogy-solving method described above makes sense.

Solution: If arbitrary vectors in Euclidian word-embedding space actually represented semantic features, then the above vector arithmetic would correspond to "subtracting out" one semantic feature and replacing it with another. On this (clearly oversimplified) view, the difference between "woman"($=w_2$) and "man"($=w_1$), for example, is a semantic gender-difference feature; if one computes the difference vector between the two ($e(w_2) - e(w_1)$) one has got a Euclidean-vector representation of the female/male gender difference, and if one adds it to some other male-gender word w_3 one "should" get the female-gender "version" of that word.

As we will see, that doesn't always work out quite so cleanly, but in some sense it is rather impressive that it works *at all*.

(b) Write a function to calculate y. The output should be a semantic vector.

Solution:

```
print(analogy("france","paris","england",vectors))
print(analogy("man","woman","king",vectors))
print(analogy("tall","taller","warm",vectors))
print(analogy("tall","short","england",vectors))
```

(c) Write a function to find the nearest words to y in terms of cosine similarity, and output the top 5.

(d) Report the top 5 results for the following analogies, and describe whether you think they are sensible and why:

i. France : Paris :: England : x

ii. man : woman :: king : x

iii. tall : taller :: warm : x

iv. tall : short :: warm : x

Solution:

```
print(analogy("singing","sang","teaching",vectors))
print(analogy("duck","ducks","pig",vectors))
print(analogy("arrive","entrance","leave",vectors))
print(analogy("cow","milk","chicken",vectors))
print(analogy("fish","water","mammal",vectors))
```

?? PythonTeX ??

One thing we immediately notice is that we tend to get the top $x = w_3$ and we have to set that aside and use the second-most-similar word to the difference vector as our candidate for x. Once we do that, the first three examples work great. The last example doesn't work well, but at the same time it's not clear what an appropriate analogy word would be.

v. Come up with 4 more analogies, 2 of which work in your opinion, and 2 of which do not work.

Solution: **?? PythonTeX ???? PythonTeX ??**

The first two of these work reasonably well. These involve inflectional morphology and the grammatical patterns supporting the analogy are likely to be manifested in words in the immediate vicinity (*a* versus *several*, or *is* versus *has*).

The last three of these are more semantically oriented analogies and they don't work so well. (In my view, the best choices would be *exit*, *egg(s)*, and *land* or *water* respectively.) I think this just reflects the fact that these models don't extract that much in the way of real word meaning after all!

(e) Did you notice any patterns or generalizations while exploring possible analogies? For the ones that went wrong, why do you think they went wrong?

References

Baayen, R. H. (2007). The `LanguageR` r package [Available at https://cran.r-project.org/package=languageR

Bresnan, J., Cueni, A., Nikitina, T., & Baayen, H. (2007). Predicting the dative alternation. In G. Boume, I. Kraemer, & J. Zwarts (Eds.), *Cognitive foundations of interpretation* (pp. 69–95). Amsterdam: Royal Netherlands Academy of Science.

Morgan, E., & Levy, R. (2015). Modeling idiosyncratic preferences: How generative knowledge and expression frequency jointly determine language structure, In *Proceedings of the 37th annual meeting of the Cognitive Science Society*.

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation, In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.