Validating Annotations in VerbCorner using Mixture Models

Phillip Hege

MIT
hege@mit.edu

Abstract

VerbCorner is an online linguistics project which works to figure out what verbs mean using semantic structure and further understand how we communicate with one another. Figuring out what verbs mean is a large undertaking and requires many annotations to be made. VerbCorner takes the approach of presenting a short story to go along with a question at the end asking about various types of meaning. With the help of many people, the task of annotating verbs is a lot easier; but then we are left with the task of validating the annotations. To evaluate the quality of the annotations we apply an item-response model described in Hovy's paper entitled Learning Whom to Trust with MACE[1]. The system that learns in an unsupervised way to do two things. 1) Identify which annotators are trustworthy and 2) predict the correct labels. We implemented a basic version of the model they described using probabilistic programming language Venture. Then we had the model run this model on the crowd sourced annotations on the semantic structure of verbs generated by users on VerbCorner. We tested the model in different conditions in hopes of finding out more about our data. Further refinement of the model, including taking a look at the priors distribution across the answers will have to be reconsidered.

Introduction

VerbCorner is providing a way to divide the task of annotating the semantic structure of verbs between many people. The task of making all of those annotations is actually very difficult. Annotating typically involves some training and a lot of time. Another approach is to instead of giving the work to a small team, distribute the questions across a larger population of people and present the annotations in a manner such as a game or quiz format that participants won't need training in order to complete. Continuing in this line of thinking, we run into issues in validating user input. Not only are we trying to pull out the right annotations but now we have the added obstacle of filtering through the noise that is all of the incorrect annotations.

We looked into many solutions for this task and found a simple approach by implementing a simple mixture model described in Hovy's paper (cite Hovy here). It made sense for our implementation because of the nature of our data. Our data, like theirs, comes from some anonymous annotators and we must evaluate the quality of the annotators as well as pull the correct annotation for each item. However, we also know the users native language, which may give us a prior belief about the quality of the annotators input. We may assume that perhaps native English speakers will be better at the task than non-native English speakers.

Model

We apply a simplified generative model provided by Hovy. This model makes the assumption that the annotator will always produce the correct label when he/she tries to. This works well in our case because our users are not compensated for their annotations. As described by Hovy in his

for $i = 1 \dots N$:
  $T_i \sim$ Uniform
  for $j = 1 \dots M$:
    $S_{ij} \sim$ Bernoulli$(1 - \theta_j)$
    if $S_{ij} = 0$:
      $A_{ij} = T_i$
    else :
      $A_{ij} \sim$ Multinomial(ξ_j)

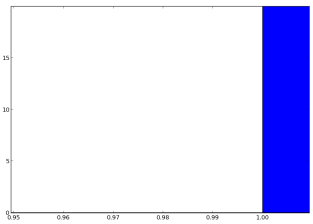
Figure 1: Generative Process N=items M = annotators T = True Label A = Annotation

paper, "*First, for each instance i, we sample the true label T_i from a uniform prior. Then, for each annotator j we draw a binary variable S_{ij} from a Bernoulli distribution with parameter $1 - j$. S_{ij} represents whether or not annotator j is spamming on instance i. We assume that when an annotator is not spamming on an instance, i.e. S_{ij} = 0, he just copies the true label to produce annotation A_{ij}. If S_{ij} = 1, we say that the annotator is spamming on the current instance, and A_{ij} is sampled from a multinomial with parameter vector j. Note that in this case the annotation A_{ij} does not depend on the true label T_i. The annotations A_{ij}*

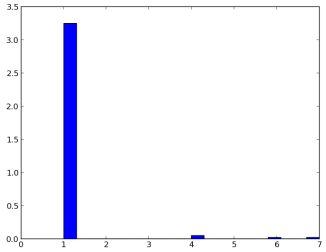
are observed while the true labels T_i and the spamming indicators S_{ij} are unobserved. The model parameters are j and i. j specifies the probability of trustworthiness for annotator j (i.e. the probability that he is not spamming on any given instance). The learned value of j will prove useful later when we try to identify reliable annotators. The vector j determines how annotator j behaves when he is spamming. An annotator can produce the correct answer even while spamming, but this can happen only by chance since the annotator must use the same multinomial parameters j across all instances. This means that we only learn annotator biases that are not correlated with the correct label, e.g., the strategy of the spammer who always chooses a certain label. This contrasts with previous work where additional parameters are used to model the biases that even dutiful annotators exhibit. Note that an annotator can also choose not to answer, which we can naturally accommodate because the model is generative. We enhance our generative model by adding Beta and Dirichlet priors on j and i respectively which allows us to incorporate prior beliefs about our annotators.".[1] Unlike Hovy's implementation our annotators are unable to choose to not answer a question, so we do not worry about that inclusion to the generative model. We implemented this generative model in Venture a probabilistic programming language currently being developed at MIT. The syntax of the code used to generate the model is attached at the end of this document.

Experiments

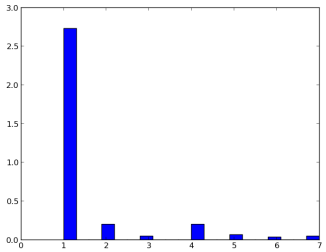
Our first task was to verify the validity of our model. In order to put it through its paces, we ran various kinds of observations on the data and then made our own prediction of what the result should be and compared it to the prediction the model generated. Here are some of the tests we ran on the data. We tried a variety of tests that could be separated into the four following categories. 1) consensus. All of the users say the same answer. 2) majority agree, others guess 3) few agree while the others guess 4) people are split between two options. The syntax used to generate the observations is attached.



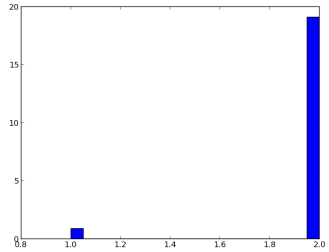
Everyone is in consensus. It is obvious to see that the model has no problem realizing the right label regardless of the trustworthiness of the users. We can see without question that this one is the correct label.



This is an case where most people say various answers but one result has some consensus around it. This result we can be sure is correct.



Even though there are guesses across the board, the model chose the one with the most consensus. This one we can be pretty sure is correct.



The confidence is shared between two options, but quickly weights to one side even with one extra annotator, perhaps more annotations need to be made on questions like this.

After confirming the model, we were ready to apply the analysis on real VerbCorner data. VerbCorner has a set of games that belong to it, each with their own story line and meaning they are testing with the same grammatical structures. The game we chose to test for this paper was the game "Good World". The data presents a storyline that ends with a statement representing particular grammatical structures and asks if this action was good, bad, neutral, depends on more context, can't tell because the verb has multiple meanings, the sentence is ungrammatical, I do not know the verb or other. A user selects one of the answers and continues to the next question. Unlike the users of MACE, our users have two key differences. 1) Our users are not com-

pensated, their participation is completely voluntary. That means they would much less likely be spamming our poll. They will most likely be putting in a good-faith effort into their answers. 2) We know our users native language, which for a task about linguistics will change people's perception of ease and accuracy when answering questions about language.

VerbCorner Data

With our data from VerbCorner we can hope to determine four things. 1) We can compare to majority rules. Are our values resulting the same as if we were to just take the majority? If so, then there is actually no real reason to even use the model. 2) Do English native speakers actually perform better? If not, then we can stop asking that question and continue collecting data regardless. 3) Are anonymous users better or worse at answering the questions? Can we trust their answers? If we cannot, then there is really no sense in us keeping track of their annotations. 4) Some questions have been posed to multiple annotators, is this necessary? What is the ideal number of annotators that have to look at the questions in order for the question to label to considered verified and for the model to get a sense of who is spamming or not.

Results

Our results thus far show some differences between what majority of people say and what the model generated. This is interesting because this may point to some validity in the model. And now we know that just because if multiple users who were labeled as 'spammers' annotated a question, that does not build confidence in the answer. This can mean we can get more accurate results with fewer annotations per item because all annotator's opinions won't be valued equally. The annotators who have consensus driven right answers will have an opinion that will be build confidence more quickly. Because of how large the dataset is, we were only able to look into a third of the total data we wanted to run. Out of 5,170 data points tested, our model generated a different label 3,072 times. That is almost 60% of the labels we gather are actually incorrect due to spammers. That number is larger than expected. We can even break down the number of responses that each system returned.

bad	2171
depends	123
good	1001
know	21
meaning	7
neutral	1779
ungrammatical	68

The distribution of answers the majority of people answered. The results are not one particular annotator, but the count of answers if you were to take the majority's opinion for the answer to each question.

bad	1154
depends	553
good	758
know	461
meaning	471
neutral	1249
ungrammatical	524

The distribution of answers the model generated.

As we had mentioned prior some answers are elicited more than others. The model did in fact start to pick up on that and increased the usage of good,bad,neutral. But as you can see, human responses rarely ever use anything other than good, bad, neutral. In fact, humans used those responses less than 300 times for 5,000 items. This makes it seem that the model's priors are not calibrated properly.

Problems with the model

Although the model seems to learn who's opinions to trust rather quickly, because we know the nature of the data, we can tell it's not as accurate as we like it to be. For one, because we know annotations will most likely be good, bad or neutral and not many will actually be any of the other answers, those answers will actually be far less likely to be selected then the first three options. Perhaps this is due to an unrealistic prior distribution or when we pulled the random result for a 'spammer' we put too much probability for some unlikely answers. Instead we should pull from results that are more likely to be selected. We know some of our results will be more probably picked because they are significant labels whereas the others are present more for completeness. Another problem with the model is that it does not address hard questions. For instance if an annotation made is based on misconceived knowledge, it could be annotated in one manner by the majority. The individuals who gave the correct annotation would receive a lower rating for questions like this because the majority believed the other way.

Future Works

Due to the size of the data set, running the model turned into a large undertaking for the computer hardware we had. First order of business will be to run the next iteration of the model on a larger computer or network of computers. With that extra power we would re run the model, looking into comparing against native English speakers, anonymous users and can look at over-sampled sentences.

```
#model defintion
def createModel(venture, lengthOfResponseList):
        venture.assume("participant_theta", """
                (mem (lambda (participant)
                        (beta 1 10)))""")

        venture.assume("item_correctResponse", """
                (mem (lambda (item)
                        (uniform_discrete 1"""+str(lengthOfResponseList)+""")))""")

        venture.assume("f", "(make_sym_dir_mult 10 "+str(lengthOfRersponseList)+")")

        venture.assume("spam_response","""
                (lambda (participant)
                  (f))""")

        venture.assume("drawTrialResponse", """
                (mem (lambda (item participant)
                        (if (flip (- 1 (participant_theta participant)))
                            (item_correctResponse item)
                            (spam_response participant))))""")


        venture.assume("noisy_true","(lambda (pred noise) (flip (if pred 1.0 noise)))")
#Test model
def testModel(venture):
    def posterior_samples(var_name,no_samples,int_mh):
        s=[];
        for sample in range(no_samples):
            v.infer(int_mh)
            s.append(v.predict(var_name))
        return s
    #everybody agrees
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item1) (quote part1)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item1) (quote part2)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item1) (quote part3)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item1) (quote part4)) 1) .2)','true')
    is_item_correctResponse = posterior_samples ('(item_correctResponse (quote item1))',no_samples=200,int_mh=300)

    #half agree, half guess
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item2) (quote part1)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item2) (quote part2)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item2) (quote part3)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item2) (quote part4)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item2) (quote part5)) 0) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item2) (quote part6)) 2) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item2) (quote part7)) 3) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item2) (quote part8)) 4) .2)','true')
    is_item_correctResponse = posterior_samples ('(item_correctResponse (quote item2))',no_samples=200,int_mh=300)

    #few agree, most guess
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part1)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part2)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part3)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part4)) 5) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part5)) 0) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part6)) 2) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part7)) 3) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part8)) 4) .2)','true')
    is_item_correctResponse = posterior_samples ('(item_correctResponse (quote item3))',no_samples=200,int_mh=300)

    #few agree, others agree
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part1)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part2)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part3)) 1) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part4)) 2) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part5)) 2) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part6)) 2) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part7)) 2) .2)','true')
    venture.observe('(noisy_true (eq (drawTrialResponse (quote item3) (quote part8)) 2) .2)','true')
    is_item_correctResponse = posterior_samples ('(item_correctResponse (quote item3))',no_samples=200,int_mh=300)
```

References

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. Learning whom to trust with mace. In *Proceedings of NAACL-HLT*, pages 1120–1130, 2013.