Extracting Commonsense Properties from Embeddings with Limited Human Guidance

Project Report - Disha Jindal

1 Introduction

This paper focuses on extracting commonsense knowledge specifically object-property comparisons of the form "cantaloupe is more round than a hammer" from pre-trained embeddings, which is challenging majorly because of the reporting bias. The major contributions of this paper are: a new approach to calculate the relative comparisons of object properties, its extension to zero shot learning settings, and achieving better results by using substantially less manually-annotated knowledge. Another useful contribution is the new data set it provided called Property Common Sense (32 properties and 689 objects). They also explored various active learning approaches and claimed the effectiveness of synthesis based approach in further boosting the accuracy of the model.

2 Model

The model takes two objects as inputs (e.g. elephant and tiger) and classifies the relation between them in either three or four categories depending upon the chosen configuration. The architecture is inspired by the idea that it compares a projection of embeddings of each of two objects to the embeddings of the target dimension of comparison trained such that the projected object embeddings are closer to the appropriate pole. Section 2.1 and 2.2 are the two main models proposed to solve this classification task.

2.1 Three-way-Model(3 way classification)

The model takes two objects and three properties corresponding to $<$, $>$, $\approx$ relations as the input and classifies the object relation in one of them. Formally,

$$P(L = s \mid Q) = softmax(R_s \cdot \sigma((X \oplus Y)W))$$

Layers: Embedding(Size: Depends upon the pre-trained embeddings used), Linear(2*EmbSize, EmbSize), Dropout(0.5)

Procedure: After finding the index of all 5 input attributes, they are fed into the embedding layer to get the distributional representation of the words. The two object vectors are concatenated and projected using the Linear layer out of which 50 percent entries are dropped using the dropout layer. The Similarity of the resulting projected vector is calculated with the projections of the three properties and the one with the maximum relatedness is chosen to be the matching class.

For training, I used Cross Entropy as the loss function, Adam as the optimizer and trained the model for 800 epochs.

2.2 Four-way-Model(4 way classification)

Since a lot of object pairs are not even comparable under the given relation, the paper caters to this requirement and extended the 3 way classification task to include an additional relation N/A. Formally it adds following additional component to the 3 way model,

$$h_x = \sigma(XW_a), h_y = \sigma(YW_a)$$

$$A_i = h_i \cdot R_> + h_i \cdot R_<$$

$$P(L = (N/A)|Q) \propto exp(A_x + A_y)$$

Layers: Embedding(Size: Depends upon the pre-trained embeddings used), Linear(2*EmbSize, EmbSize), Linear(EmbSize, EmbSize), Dropout(0.5)

Procedure: Along with following the procedure mentioned in the 3 way model, both object vectors are projected using another Linear Layer following which 50 percent entries are dropped using the dropout layer. Then, their similarity is calculated with both opposing poles and added together. And then, both projected object vectors are added together and this is used as a proxy for N/A. Out of these four classes, the one with

the maximum relatedness is chosen to be the matching class.

Along with these two models, the authors tried a couple of variations around the model:

2.3 PCE(one-pole)

In PCE(one-pole), label representations are generated from just a single property embedding $R_<$ and others are derived from it by adding an encoding layer. Formally,

$$R_{\approx} = \sigma(R_< W_2), R_> = \sigma(R_< W_3)$$

Layers: Embedding(Size: Depends upon the pretrained embeddings used), Linear(2*EmbSize, EmbSize), Linear(EmbSize, EmbSize), Linear(EmbSize, EmbSize), Dropout(0.5)
Procedure: The architecture and procedure for this is almost same as that of 3 way apart from the way the embeddings are calculated for the two relations. Rather than using the embeddign layer to get the word representations of all five attributions, embedding layer is used for 3 attributes(2 objects and 1 relation) and the other 2(relations) are calculated using the linear projections of the first relation.

2.4 PCE(no-reverse)

In the 3 way model, the author leveraged the fact that $A > B$ is equivalent to $B < A$ at the test time by feeding a reversed object pair to the network as well, and taking the average of the two outputs before the softmax layer to reduce prediction variance. Model without this technique is referred to as PCE(no-reverse).
Layers: Embedding(Size: Depends upon the pretrained embeddings used), Linear(2*EmbSize, EmbSize), Dropout(0.5)
Procedure: Keeping the architecture and the training procedure identical, the only difference between this and 3 way model lies in the evaluation phase. In no reverse, only direct tuples are fed to the model whereas in 3 way model both direct and reversed object pair are fed and are averaged to smoothen the output.
Since, above four models share most of the logic, I created one model for all of them during implementation and handled the variations using flags.

2.5 Emb Similarity

To reproduce the baseline, I modelled the embeddding similarity model mentioned in the paper.
Layer: Embedding(Size: Depends upon the pretrained embeddings used)
Procedure: After finding the index of all 5 input attributes, they are fed into the embedding layer to get the distributional representation. Then, I found the cosine similarity of vector: $obj1 - obj2$ with all the relation vectors and for prediction, chose the one with the maximum similarity.

2.6 Majority

I also reproduced the majority baselines for Table 1 and Table 3. It was done by finding the most frequent class and comparing it with the labels.

3 Dataset

The authors used VERB PHYSICS data set and also developed their own PROPERTY COMMON SENSE data set. For the word embeddings, they used 3 pre-trained models. Following subsections give a detailed description of these datasets.

3.1 Embedding Data

Word2vec: Embedding dimension: 300; Trained on: 6B tokens; Size of word to Index map: 923; Size of index to embedding map: (924, 300) with no word corresponding to index 722.
GloVe: Embedding dimension: 300; Trained on: 100B tokens; Size of word to Index map: 834; Size of index to embedding map: (835, 300) with no word corresponding to index 633.
LSTM: Embedding dimension: 1024; Trained on: 1B LSTM language model; Size of word to Index map: 894; Size of index to embedding map: (895, 1024) with no word corresponding to index 719.

3.2 Verb Physics

Attributes of the dataset:
Objects: 200; ***Properties***: 5 (big, fast, heavy, rigid, strong); ***Training Data***: 587; ***Dev Data***: 5418; ***Test Data***: 6007
There are a number of issues with this dataset:
1.More than half of the dataset contains non comparable triples. 2. Embeddings are missing for some of the objects and properties. Specifically, 10 records from training data could not be used due to missing words in all of the embeddings: Word2Vec, Glove and LSTM.

3.3 PCE

Attributes of the dataset:
Objects: 689(Extracted from McRae Feature

Norms dataset); ***Properties***: 32; ***Object-Property triples***: 3148; ***Training Data***: 1740; ***Test Data***: 1568

Issues with the dataset:

The problem 2 mentioned in the verb physics dataset is even more severe in the case of PCE. Following is the number of invalid data entries because of the missing embedding of either one of the objects or the relational property:

LSTM: Training: 136 Test: 115

GLove: Training: 390 Test: 346

Word2Vec: Training: 3 Test: 3

A large number of these seem like spelling mistakes. So, a cleanup of data followed by enriching the embedding dataset would lead to better usage of both the datasets.

4 Experimentation details

Following are the major implementation components:

Data Preparation 1. Get pre-trained embedding weight matrices (Word2Vec, GLove and LSTM in this case). 2. Create a word to index dictionary from the embedding data. 3. Filter the training and test data by removing entries for which there is no matching word embedding available. 4. Format the data in quintuples of the form $< Obj_1, Obj_2, Rel_1, Similar, Rel_2 >$ containing the corresponding word indices from the dictionary.

Models Section 2 describes all the models in detail. Create all of the mentioned variations of the models.

Training and Evaluation Metric Decide on the loss function(Cross Entropy), evaluation metric(Accuracy), optimizer(Adam), epochs etc. Use development dataset to come up with the most appropriate set of hyper parameters. Although for the given setting, I could reproduce the mentioned results using the parameters mentioned in the paper.

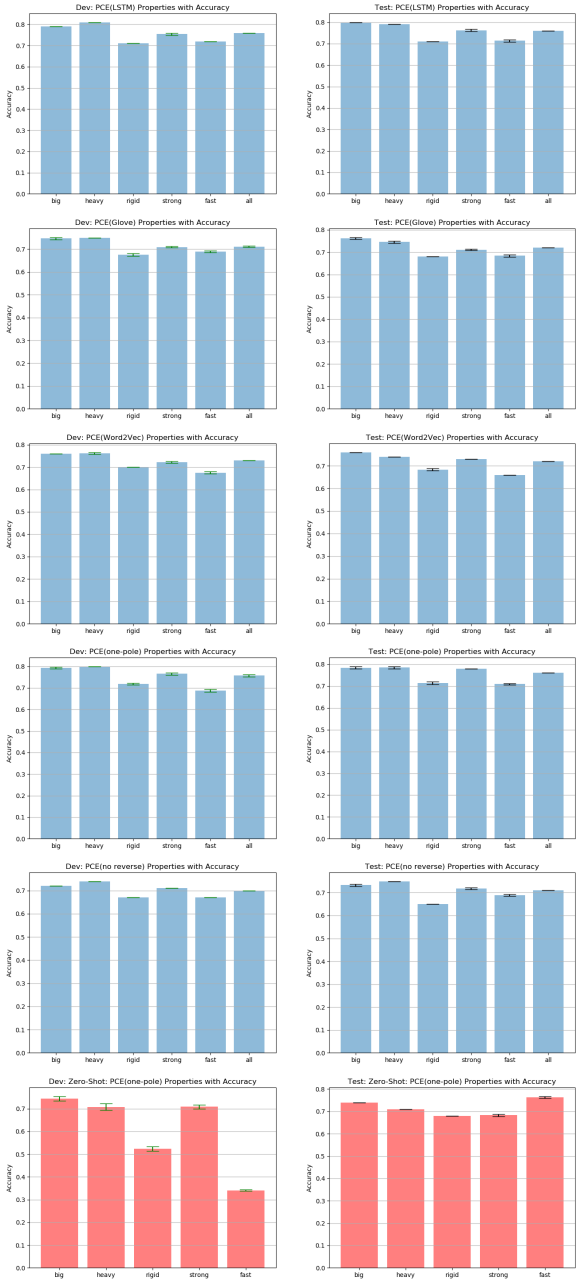
Main Script 1. The main script parses the arguments 2. Loads dictionary based on mentioned embedding 3. Prepares the matching dataset and filter it based on reverse, relation, zero shot etc. 4. Instantiates the model 5. Trains the model 6. Evaluates on dev set and test sets.

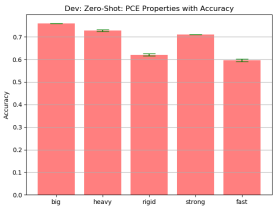
Stastistics 1. I wrote shell scripts each for Table1, Table2 and Table3 to run the main script with multiple configurations for multiple runs which dump the results in a file. 2. Wrote three corresponding scipts to which take the results from the file, calculate mean and variance of various runs and plot the results.

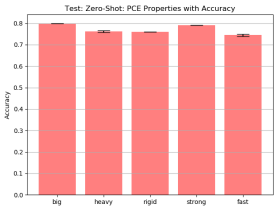
5 Results

5.1 Comparison against results from the paper

I reproduced all the results in three tables from the paper and ran each of the configurations 10 times. Post which, calculated mean and variances, and plotted them with the error bars. Tables 1, 2 and 3 show the results in the paper along with the reproduced(reporting the mean of 10 runs) results







5.2 Analysis

As we see from the plots above that the reproduced results are very similar to the results mentioned in the paper. Also, the standard deviation of various runs is of the order of 0.0017.

It is also quite evident that performance of PCE(LSTM) is better than Majority class, Forbes and Choi's approach as well as other PCE variants such as PCE(one-pole), PCE(no reverse), PCE(Glove) and PCE(word2vec).

To establish that the results are not coincidental, I performed some statistical tests on the obtained results. Roughly formulating the hypothesis testing setting:
Alg_1: Algorithm 1
Alg_2: Algorithm 2
M: Accuracy
H_0: M(Alg_1) - M(Alg_2) ≈ 0
H_1: M(Alg_1) - M(Alg_2) > 0
This is a classification settings with the number of classes as 4. The observations are categorical in nature so Student's t-test or Wilcoxon signed-rank tests are not the right fit for this problem. I then chose **McNemar's test** for this setting as the data points(instance wise accuracy i.e. 1 if the prediction is correct and 0 otherwise) are binary labels and conducted analysis over various combinations. Two algorithm variants are listed along with the p-values from the test

Analysis of Table 1 results
Algorithm1 vs Algorithm2: p-value
1. PCE(LSTM) vs Majority: 6.78e-198
2. PCE(Glove) vs Majority: 7.0436e-140
3. PCE(word2vec) vs Majority: 1.86e-135
4. PCE(LSTM) vs PCE(Glove): 3.39e-16
5. PCE(LSTM) vs PCE(word2vec): 4.4e-19
6. PCE(LSTM) vs PCE(noreverse): 1.33e-19
7. PCE(LSTM) vs PCE(one-pole): 0.552

It is clear from the points 1,2 and 3 above that all PCE variants are way better than the majority baseline which is expected. Points 4 and 5 show the significance of LSTM over word2vec and glove. Point 6 supports the importance of reverse tuples in evaluation and point 7 shows that there isn't significant difference between PCE(LSTM) vs PCE(one-pole) which again corroborates the author's claims.

Analysis of Table 2 results (property- size)
Algorithm1 vs Algorithm2: p-value
1. PCE(LSTM) vs Emb-Similarity: 9.81e-42
2. PCE(LSTM) vs PCE(one-pole): 0.510

Point 1 shows clear significance of PCE(LSTM) over Emb-Similarity baseline results which just finds the similarity between the embeddings without any projections and training to separate the objects farther in the vector space. Point 2 when compared to point 7 from the analysis of table 1 shows the relatively more significance of one-pole in zero shot learning setting.

5.3 Example predictions from the model

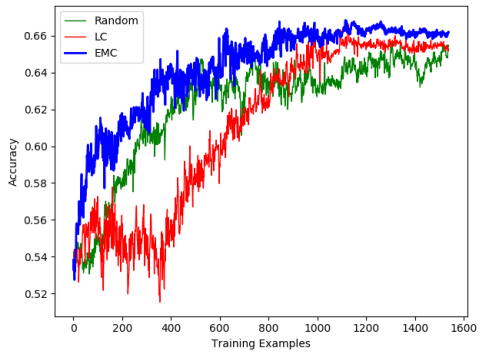
Tables 1, 2 and 3 give the detailed accuracy results. For instance: Given a quintuple of the form (lion, hamster, short, similar, tall) it returns 1, which means that lion is taller than hamster.

5.4 Additional experiments

I developed the following active learning settings and used Word2Vec embeddings for all of them. For the warmup, trained all the models on 200 random training examples. All of these experiments are performed on PCE dataset because it provides an option to categorize a query as N/A.
Random Pool Based approach After warming up the model, ran the experiment for 1540 iterations and picked a training example randomly from the leftover entries each time and trained the model for 20 more epochs and recorded the accuracy at each iteration.
LC (Least Confident) Pool Based approach In each iteration of this approach, every data entry in the leftover dataset is evaluated and the one with minimum confidence in its selection is added to the dataset and the model is then trained for 20 more epochs.
EMC(Expected Maximum Change) Pool Based AL In each iteration of this approach, I trained the model with each individual input and calculated the sum of gradient change it lead to. The input which lead to the maximum gradient change was selected and added to the training pool.
Sampling based synthesis AL strategy This

approach is different from the above discussed pool based approaches as query is synthesized in this unlike others. In PCE dataset, there are 689 objects and 32 properties. In each iteration, I randomly selected 2 objects for each property and constructed a dataset with these 32 quintuple queries. Then, I train the model with these and choose the query which is most uncertain similar to the Least Confident approach discussed above. It then asks the user(oracle) to give the label to the synthesized query.

Analysis I ran the first three approaches for the entire dataset(1540 iterations after removing 200 for warmup) and following is the visualization of the number of training examples vs the accuracy. We can see that EMC has the fastest convergence rate among these three.

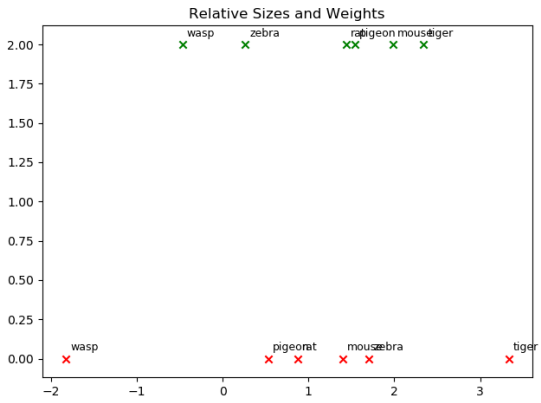


I also tried synthesis based approach for few iterations which showed even faster convergence rate but due to its high labelling cost it was in-feasible to include it for 1540 iterations.

Relative Property Analysis Along with Active Learning, I did some experimentation around analyzing the projected object vectors and its relatedness to various relation pairs.

I chose following 6 animals/birds from the object list: [***tiger, zebra, rat, wasp, pigeon, mouse***] After getting the word representations of all of them, I did the property vector subtraction: big-small and heavy-light. Then, calculated the similarity of each one of the objects with these two property vectors to find the relative big(ness) and heavy(ness). Following is the visual representation: Red Markers correspond to Size(Big-Small) and Green Markers correspond to weight(Heavy-Light)

Objects are better separated in the size domain as compared to the weight domain. The relative scores in this sample are mostly sound except zebra scored lighter than rat and mouse.



6 Problems Faced

The paper talks vaguely about the entire dataset preparation process: there is no mention of the selection criterion for the properties, broad categories covered or the process of arriving at final set of adjectives for each category. But from implementation point of view, the data-set provided was in proper format and did not require much processing apart from misspelled and missing data which required data validation and filtering.

The procedure employed for various baselines(Emb-Similarity, Majority) was not clear which required guessing the intent. From reproducing perspective, I found the Model Interpretation and pole sensitivity parts to be under-specified too.

7 Conclusions

The results mentioned in the paper seem legitimate and were reasonably easy to reproduce. Also, the data-set provided was clean but the quality is not upto the mark as a large number of queries in verb physics are not quite informative. Also, a large number of object words from the PCE data-set are either misspelled or are missing in the provided word embeddings.

It is also interesting to see that the results of the fully trained model are pretty robust as the accuracies remained almost consistent during PCE(one-pole) training as well as PCE(no-reverse). Whereas, zero-shot learning results are highly sensitive to different variants. The paper lacks clarity at different points and does not support the results using any significance analysis. But nevertheless, it is a step towards automating the commonsense extraction process. Also, after doing the significance analysis, I figured that the given results are indeed significant.

Model	Development						Test					
	big	**heavy**	**stren**	**rigid**	**fast**	**all**	**big**	**heavy**	**stren**	**rigid**	**fast**	**all**
Paper Results												
Majority	0.50	0.54	0.51	0.50	0.53	0.51	0.51	0.55	0.52	0.49	0.50	0.51
PCE-LSTM	0.79	0.81	0.75	0.71	0.72	0.76	0.80	0.79	0.76	0.71	0.71	0.76
PCE-GloVe	0.75	0.75	0.71	0.67	0.69	0.71	0.76	0.75	0.71	0.68	0.68	0.72
PCE-Word2vec	0.76	0.76	0.73	0.70	0.68	0.73	0.76	0.76	0.73	0.68	0.66	0.72
PCE(one-pole)	0.80	0.81	0.77	0.65	0.72	0.75	0.79	0.79	0.77	0.65	0.72	0.75
PCE(noreverse)	0.72	0.74	0.71	0.67	0.67	0.70	0.73	0.75	0.72	0.65	0.68	0.71
Reproduced Results												
Majority	0.50	0.54	0.51	0.50	0.53	0.49	0.51	0.55	0.52	0.49	0.50	0.49
PCE-LSTM	0.79	0.81	0.75	0.71	0.72	0.76	0.80	0.79	0.76	0.71	0.71	0.76
PCE-GloVe	0.75	0.75	0.71	0.68	0.69	0.71	0.76	0.74	0.71	0.68	0.68	0.72
PCE-Word2vec	0.76	0.76	0.72	0.70	0.68	0.73	0.76	0.74	0.73	0.68	0.66	0.72
PCE(one-pole)	0.79	0.80	0.77	0.72	0.69	0.76	0.78	0.79	0.78	0.71	0.71	0.76
PCE(noreverse)	0.72	0.74	0.71	0.67	0.67	0.7	0.73	0.75	0.72	0.65	0.69	0.71

Table 1: Comparison of results on the VERB PHYSICS dataset

Model	Development					Test				
	big	**heavy**	**stren**	**rigid**	**fast**	**big**	**heavy**	**stren**	**rigid**	**fast**
Paper Results										
Random	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33
Emb-Similarity	0.43	0.55	0.51	0.43	0.35	0.37	0.53	0.48	0.43	0.35
PCE(one-pole)	0.73	0.71	0.67	0.53	0.34	0.74	0.72	0.68	0.53	0.32
PCE	0.76	0.72	0.71	0.62	0.60	0.74	0.73	0.70	0.62	0.58
Reproduced Results										
Random	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33
Emb-Similarity	0.43	0.55	0.50	0.44	0.40	0.37	0.53	0.48	0.44	0.38
PCE(one-pole)	0.74	0.71	0.71	0.52	0.34	0.74	0.71	0.72	0.52	0.34
PCE	0.76	0.73	0.71	0.62	0.60	0.74	0.73	0.70	0.62	0.58

Table 2: Comparison of zero shot results on the VERB PHYSICS dataset

Model	**Test(Paper)**	**Test(Repro)**
Random	0.25	0.25
Majority	0.51	0.32
PCE(GloVe)	0.63	0.71
PCE(Word2vec)	0.67	0.67
PCE(LSTM)	0.67	0.693

Table 3: Comparison of four way results on the PCE Dataset