Predicting Perception Uncertainty for Aspect-Based Sentiment Analysis

Final Report

Guided Research
Department of Informatics, Technical University of Munich (TUM)

Advisor	Gerhard Hagerer, M.Sc
Supervisor	PD Dr. Georg Groh
	Research Group Social Computing
Authors	Farrukh Mushtaq
Date	Munich, February 3, 2020

Contents

1 Introduction

The main tasks undertaken for this guided research are as follows:

1. Performing Aspect-Based Sentiment Analysis (ABSA) using transformer based XLNet[1] model.
2. Developing a neural network model to predict perception uncertainty for a given sample.

In this report, I mainly go into details about the experiments and their results, according to our research goal.

2 Methodology

Keeping in mind the recent progress in natural language processing, we have used deep pre-trained neural networks to obtain meaningful latent representations of text. These representations are then being used in various ways to perform required downstream tasks. The pre-trained models and the dataset being used in this work are detailed below.

2.1 XLNet

XLNet[1] model is inspired by the Transformer[2] architecture and improves upon the state-of-the-art performance by BERT[3]. BERT showed state-of-the-art performance compared to the previous unidirectional models because it was able to learn from / attend to all the words in a sentence using bidirectional context learning. A major flaw in BERT however is the discrepancy between pretraining and finetuning steps of BERT due to the data masking which is introduced at pretraining but do not exist at finetuning step. Moreover there's an independence assumption between every masked token reconstruction. XLNet, on the other hand, employs a generalized autoregressive model which naturally avoids both of BERT disadvantages while still benefiting from bidirectional learning. XLNet enables bidirectional context learning by using permutations of the autoregressive factorization order. For each sentence of length T, T ! different autoregressive permutations exist and during pre-training XLNet, we sample from all of these T ! factorization orders. Since the model parameters remain the same for all factorization orders, every word in the sentence is able to see every other word in the sentence, hence achieving the bidirectional learning objective. The factorization order is implemented using the attention mask in the architecture and the sequence order is kept the same, to avoid the same pretrain-finetune discrepancy that BERT suffers from. XLNet achieves state-of-the-art performance on 18 NLP tasks, some of which include language understanding, reading comprehension, text classification and document ranking. Just like BERT, XLNet has been pretrained on BooksCorpus[4] and English Wikipedia, along with some other datasets. There are two versions available for XLNet: XLNet-Large and XLNet-Base. For this research, I've used XLNet-Base model, which is comparable to BERT-Base model. Similar to BERT, XLNet also produces 768 dimensional vector representation corresponding to each word in the sentence together with a special *[CLS]* token, that condenses information from the complete sentence.

2.2 Universal Sentence Encoder

Universal Sentence Encoder[5] (USE) is a pre-trained neural network model that is intended to encode sentences into embedding vectors that can be used for transfer learning in other NLP tasks. Using sentence-level embeddings in USE tend to outperform word-level embeddings in transfer learning and require very small amount of supervised training data for a downstream transfer task. There are two different variants of this model with a trade-off between accuracy and computing resources. The lighter model with lower accuracy is based on Deep Averaging Network[6] (DAN), in which word embeddings for input words are averaged together and then passed through a feedforward deep neural network to produce sentence-level embeddings. Compute time for this model is linear with the length of the input sentence. The more compute-heavy model uses the encoding portion of the Transformer[2] architecture for computing sentence level embeddings and makes use of attention to take the ordering and context of all words into account. This model results into the best performance for USE and is used in this research for calculating semantic similarities – see section 4.1. Both of these variants outputs a 512 dimension embedding vector for each sentence. The data sources for unsupervised pre-training of these models include Wikipidea, web news, web question-answer pages and discussion forums.

2.3 Organic Dataset

Organic dataset is the in-house collected dataset, composed of social media comments about organic foods. It has both annotated and unannotated samples and for our tasks, I've used the annotated samples from the organic dataset. Annotated organic dataset is composed of 10441 samples, out of which 5561 are relevant to the organic domain, so we filter out to only use the relevant comments. We don't do any special preprocessing for the annotated dataset except for combining samples with multiple labels into one sample. For the ABSA labels, I'm combining sentiments with coarse entities and coarse attributes to produce a total of 54 ABSA classes.

3 ABSA using XLNet

In this section, I describe the various experiments that I did related to Aspect-Based Sentiment Analysis (ABSA) using XLNet model. The experiments were performed on the annotated organic dataset and the eventual goal of the task was to obtain the best ABSA performance on labeled organic set using superior word embeddings and different loss formulations. Below I go into detail about this whole process.

3.1 Model Architecture

The architecture defined for the model is simple and combines XLNet with linear layers (see Fig. 1). Input to the model are individual sentences from the Organic dataset, which are then fed through to the WordPiece tokenizer. This tokenizer is helpful for the XLNet model to learn better as it breaks down the words in a sample to their possible roots. As a result of this, many different grammatical variations of the same root word are learned to be similar e.g. *walking*, *walked* and *walks* are from the same root word *walk*. The resultant tokenized sample is then padded on the right with two special tokens *[SEP]* and *[CLS]* to make it a valid input sample for XLNet. This padded tokenized sample is then fed into the pretrained XLNet model. For every single token in the input, XLNet produces a 768 dimensional vector and additionally it also produces a special 768 dimensional *[CLS]* token, which stands for Classification token, which combines the information for all the tokens into one and thus can be used as a sentence embedding. This *[CLS]* token can then be used for a wide range of classification (or regression) tasks. I built a simple multi-head model on top of *[CLS]* token where each head is composed of a single linear layer. There are 3 heads in total for my model:

1. ABSA Head.
2. Coarse Entity Head.
3. Fine Entity Head.

For all heads, the base of the model is shared and they only differ in their own linear layers. Since every sample in the annotated organic dataset can have multiple labels, this problem is treated as a multi-label, multi-class classification problem. The loss function used for training the model is **Binary Cross-Entropy** (BCE). The result obtained using this model and BCE loss can be seen in Table 1. Using BCE loss, the model performs better on ABSA task when only a single head is trained, compared to when all the heads are trained together. When all heads are trained together, then we can also evaluate the performance of each individual head during test time and the corresponding results for BCE loss can also be seen in Table 1.

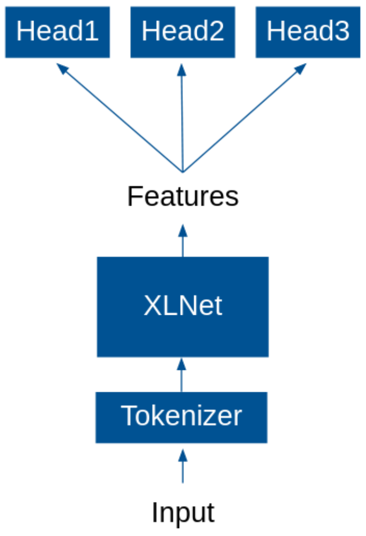


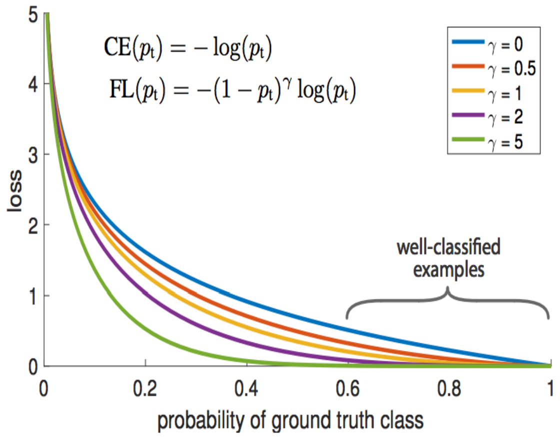
Figure 1: Model architecture defined for ABSA using XLNet.

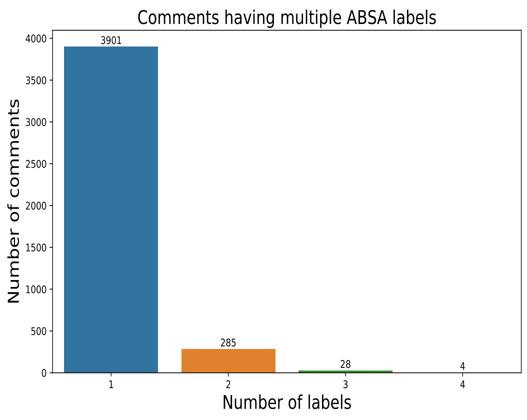
3.2 Focal Loss

Focal loss[7] is a loss criterion that was developed at Facebook AI Research (FAIR) with the primary goal of reducing the effect of standard cross-entropy in the case of vast number of easy background examples, compared to fewer relevant foreground samples. What this means in practice is that we want to focus our training on a fewer number of hard examples where our model is pretty uncertain and prevent the huge number of easier examples from overwhelming our model. This is achieved by adding a factor γ to the standard cross-entropy formulation (see Fig. 2a). Since the main problem in training our model with multi-label classification targets using BCE loss were the amount of *background* or *not present* classes for each sample, focal loss as a solution was considered. The idea was that since focal loss suppresses the effect of background classes, our model should be able to better learn about the relevant present classes. However, experimentation with the use of focal loss with our model showed that either the model did not improve or the performance actually deteriorated. The reason for this behaviour is that the problem that focal loss tries to solve and the problem that we're facing are not the same. For focal loss to work, the model must be fairly certain about the background classes so that their loss values could be suppressed. But in our case, our model is equally uncertain about both the relevant and background classes. So any value of γ affects all the classes in the same way.

3.3 Cross-Entropy Loss

Further investigation of the organic dataset showed that more than 92% of samples only contain one label (see Fig. 2b). Thus treating this dataset as a multi-label dataset does not make much sense and hence, we changed our task from a multi-label task to single-label task. I randomly selected a





(a) Focal loss compared to standard cross-entropy. Higher the value of γ, lesser is the loss for samples on which model is certain. With γ=0, focal loss becomes cross-entropy

(b) Count of labels per sample for annotated organic dataset

label from all the labels of a given sample, for those samples which had multiple labels. Just as the problem changed from multi-label to single-label, loss function is also changed from BCE to cross-entropy loss. The performance of our model with cross-entropy loss is also given in Table 1. Using cross-entropy, combined model performs slightly better on ABSA as compared to single ABSA head.

As a final experiment, I also tried weighted cross-entropy in which each sample loss is weighted by the presence of that sample class in the dataset. The performance on ABSA significantly deteriorates with weighted cross-entropy, compared to the small effects on the other heads.

Confusion matrices for the combined model trained on single-label problem with cross-entropy loss are provided in the appendix (see Appendix A) to visualize model uncertainty on any given class.

		F1 Scores (Micro)		
Loss function	Heads	ABSA	Coarse Entity	Fine Entity
Binary Cross-Entropy	All Heads	0.26	**0.88**	**0.54**
Binary Cross-Entropy	Only ABSA Head	0.31	-	-
Cross-Entropy	All Heads	**0.34**	**0.88**	0.53
Cross-Entropy	Only ABSA Head	0.33	-	-
Weighted Cross-Entropy	All Heads	0.21	0.83	0.51

Table 1: Results of ABSA using various losses and both single-head and multi-head model variations

4 Perception Uncertainty Prediction

For a text sample, perception uncertainty refers to how much a human would be uncertain about the true label of that text. The goal is to propose a model that can predict such uncertainty for any given text. Such predicted perception uncertainty can then be used in many downstream tasks like weighting the labels of a text sample with its predicted perception uncertainty, as it would give you a measure of how much belief could be put into any given human label. There are 2 sub-problems that differentiate this task from a simple regression task:

1. How to obtain the perception uncertainty targets for training the model?
2. How to evaluate the performance of the trained model?

If we can propose a way by which we can approximate "ground truth" perception uncertainty, irrespective of the scale, then our task simplifies down to training a regression model. This is where the idea of semantic similarity comes in.

4.1 Semantic Similarity

Semantic similarity refers to how much a given sample is similar in meaning to another. For calculating semantic similarity, we use the ideas from another paper[8] by our chair, in which semantic similarity is calculated based on deep pre-trained embeddings obtained using Universal Sentence Encoder (USE)[5]. It is shown in the paper that inter-rater reliability for each class can be directly calculated by comparing text embeddings mutually within and between ABSA classes. We've taken the idea of *semantic class similarities* from the above mentioned paper and applied it to our problem of calculating perception uncertainty training targets.

For perception uncertainty, experiments are performed using *coarse attributes* of the organic dataset because of its moderate number of classes and inter-class separability. For obtaining the semantic similarity, first organic dataset samples are converted into sentence embeddings using USE. Then the distance between any two sentence embeddings is calculated based on the *inner product* of two embeddings. High inner product values means that the two sentences are semantically similar and the distance between them in embedding space is low. This process is repeated for each sentence, where each sentence is compared with the current sentence. Finally semantic similarity between two classes is calculated to be the average of all the sentence similarity values from the two classes. Both inter-class and intra-class semantic similarities for coarse attribute can be seen in Fig. 3. Other than inner product, *cosine distance* and *mean squared error* are also tried as similarity metric, but the results are almost the same.

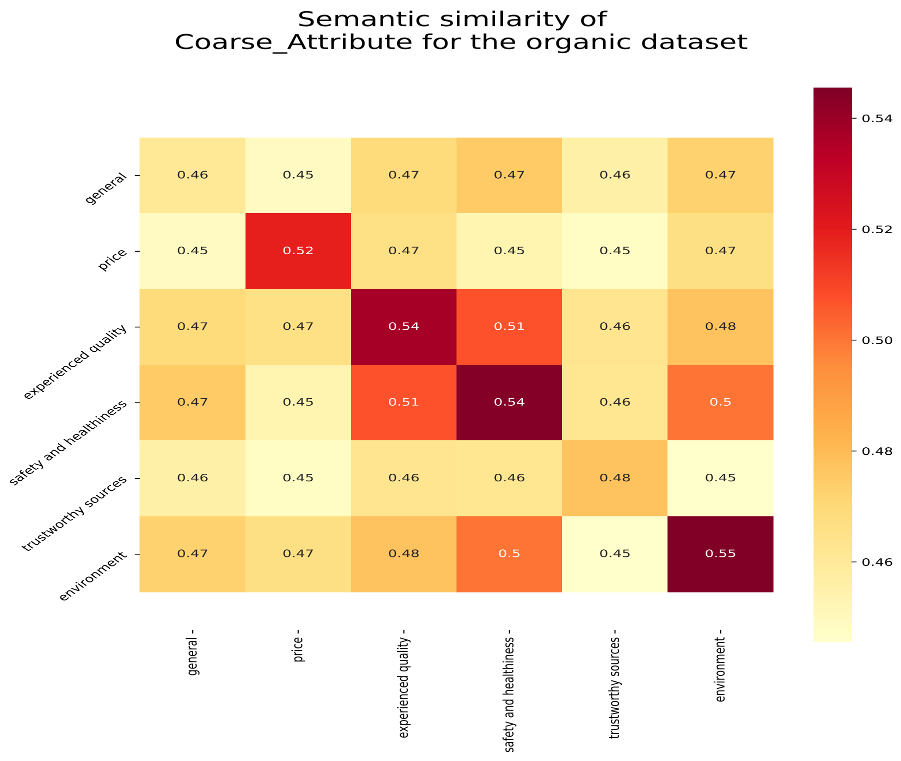


Figure 3: Semantic similarity for coarse attribute on organic dataset.

4.2 Class Centroids

Our novel idea in this research is that the closer a sample is to its own class centroid in the embedding space, the higher is the semantic similarity between the sample and class centroid and therefore, the lesser is its perception uncertainty and vice versa. For testing this idea, we need to calculate the class centroids. For this purpose, I have used a special variant of K-means algorithm called **constrained pairwise K-means** (COP-K-means)[9], which can incorporate pair-wise constraints for clustering. There are two types of possible constraints: **must-link** (both samples in the constraint must be in the same cluster) and **cannot-link** (both samples in the constraint must be in different clusters). For our case, samples belonging to the same class are involved in the must-link constraints and samples belonging to different classes are involved in the cannot-link constraints. After obtaining all the constraints, we can apply the COP-K-means algorithm to get our required class centroids. One interesting observation is that all the samples belonging to a class are involved in the must-link constraints, basically saying that all samples of the same class should be in the same cluster. Moreover, all the different class samples are in the cannot link constraints, basically saying that all samples from different classes should be in different clusters. As a result of this observation, it can be theorized that COP-K-means with such specific constraints would find cluster centroids that are just the average of the corresponding class samples. The theory is then proven through experimentation by comparing centroids found through COP-K-means with the centroids obtained by class samples averaging and showing them to be the same.

After obtaining the centroids, semantic similarity or distances of each sample from these centroids is calculated using inner product as before. Then, these semantic similarities with class centroids are combined in 3 ways to obtain 3 different types of training targets:

1. Semantic similarity between a sample and its class centroid (PU_S)
2. Sum of semantic similarities between a sample and all class centroids, each weighted by respective class-wise similarities (PU_{WS})
3. Sum of unweighted semantic similarities between a sample and all class centroids (PU_{uWS})

Afterwards all 3 targets are normalized. The distribution for all 3 training targets on the organic dataset can be seen in Appendix B. Since the distribution of values for all 3 metrics is almost the same, I chose PU_S as the metric to perform all the experiments with (Initial experiments were done for all 3 training targets, but the results were almost the same so for further experiments, only 1 training target was selected).

4.3 Model Architecture

Once we have the training targets, we can train our model on it but we need to make some changes to the classification model to make it suitable for regression task. The shared part of the model from before remains the same and in place of multi-heads, we add a single linear layer to predict the perception uncertainty value (see Appendix C). Also we now use **L1** loss to reflect the change in task category. Everything else remains the same, input goes to tokenizer, tokenized input passes through pre-trained XLNet and gives *[CLS]* token which is then used by the linear layer to predict perception uncertainty. As a final step, predicted perception uncertainty is passed through a Sigmoid layer.

4.4 Model Evaluation

After training on the organic annotated dataset, we can first test it on the organic test set, which has the desired perception uncertainty targets obtained in the same way as the training targets. But, the actual evaluation is done on a separate dataset which has multiple annotations per sample, from which "ground truth" perception uncertainty can be calculated. Majority of the samples in this *evaluation dataset* are annotated by roughly 10 annotators (Appendix D), which are enough to calculate good ground truth targets. But once domain irrelevant samples are removed, then the distribution becomes more balanced with a lot of samples having very few annotations (Appendix D).

This creates an issue as it results into lower quality ground truth perception uncertainty targets calculation.

4.4.1 Calculating Ground Truth Perception Uncertainty

I've used **Normalized Information Entropy** (NIE) to calculate ground truth targets from the evaluation dataset. As the name suggests, Its an *entropy* based metric which results into higher values for samples where annotations for the same sample differ a lot and vice versa. Normalizing this entropy metric simply moves the entropy values in the range from 0 - 1, with 0 NIE value for samples where all annotations for a sample are same and 1 NIE value for samples where all annotations for a sample are different. The equation for NIE is given in Eq. (1), where N is the total number of annotations per sample.

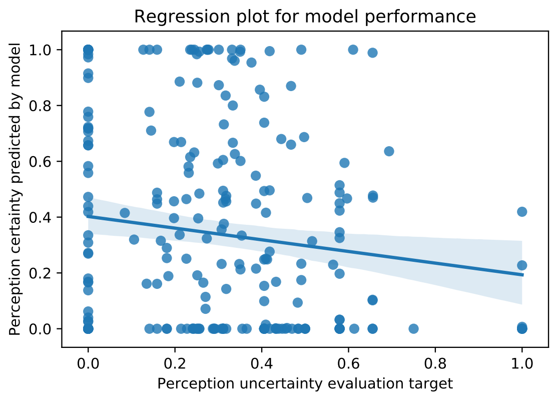
$$0 \leqslant -\sum_{i=1}^{N} p_i * log_N \, p_i \leqslant 1 \tag{1a}$$

$$p_i = \frac{1}{\mathrm{N}} \, \forall p_i \tag{1b}$$

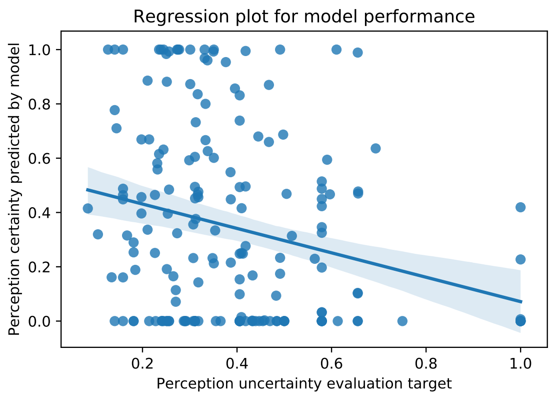
There are 2 interesting observations that can be made from the resultant NIE targets. First, majority of the coarse attribute NIE targets are 0 (Fig. E.1), which means that for most of the evaluation samples, annotations for coarse attributes is same meaning that for those samples perception uncertainty is minimal. Second observation, which nicely ties in with the first, is that if we calculate NIE targets for all possible ABSA decompositions, we can see that the number of classes for a prediction entity is directly related to the mean perception uncertainty values for that entity. This means that if we have higher number of classes for a prediction entity, NIE targets would have higher values on average as compared to a prediction entity with smaller number of classes (Fig. E.2).

4.4.2 Evaluation Results

Once we have the evaluation targets, we can use our trained model to obtain the predictions for the evaluation dataset. We use **Pearson correlation coefficient** (PC) to compare the model predictions with that of the evaluation NIE targets and evaluate our model performance. For NIE targets, the relation between the predictions and the targets needs to be negative since higher NIE value should have lower prediction value which correspond to lower semantic similarity between that sample and its centroid meaning higher perception uncertainty. When all samples from the evaluation dataset are considered, we can see that that there is a correlation but the PC value is low, see Table 2. This is due to the overwhelming effect of the huge number of evaluation samples with 0 NIE target. Once we remove the samples with 0 NIE target, we can see even higher correlation between the evaluation



(a) Regression plot for all samples



(b) Regression plot for samples with NIE values greater than 0

Figure 4: Regression plots for model performance. A steeper regression line indicates higher correlation

targets and the model predictions, showing good predictive power of the trained model. Weighted Pearson correlation is also tried in which every sample is weighted by the representation of its class in the dataset. It improves the correlation result on all samples by *0.01*. Regression plots for model performance on all evaluation samples compared with only evaluation samples with PC values > 0 is shown in Fig. 4.

Two other regression based evaluation metrics are also tried to improve the model evaluation performance. First metric is the same as NIE with the only exception being that N is fixed and equal to the maximum number of annotations for any given sample. When we use this modified definition of NIE to calculate evaluation targets, the results turn out to be completely wrong (Table 2). The correlation is high but it's a positive correlation which shows the exact opposite of what we want to achieve. For the second metric, we use the *majority class count* principle. So the more samples there are in the majority class, the less is the perception uncertainty, see Eq. (2). Using this metric turns out to produce 0 correlation between predictions and evaluation targets.

$$\text{perception uncertainty} = 1 - \frac{\text{number of annotations for majority class of sample s}}{\text{total number of annotations for sample s}} \tag{2}$$

One final evaluation metric that we tried is a combination of NIE with confidence intervals to transform the regression problem into a classification one. First, we fix the N used in NIE such that it covers majority of the data and if for any sample, the number of annotations are greater than N, then we randomly select N annotations from all given annotations. After that, we calculate a *confidence interval* for each sample; the more annotation a sample has, the lower is its confidence interval. If for any given sample, the prediction by the model falls somewhere in between $evaluationtarget \pm confidenceinterval$, then we say that the prediction is correct. This can be better seen in Eq. (3).

Evaluation Criteria	Pearson Correlation (All samples)	Pearson Correlation (Samples with NIE > 0)	Weighted Pearson Correlation
Normalized Information Entropy	**-0.14**	**-0.25**	**-0.15**
NIE with fixed N	0.28	0.28	0.17
Majority Class Count	0.00	0.00	-0.03

Table 2: Model performance using various evaluation criteria

We can get a classification accuracy of **0.23**, which is comparable to the PC score we got for NIE but since the metric doesn't have theoretical backing and is self-made based on intuition, it might be dubious for an external reviewer.

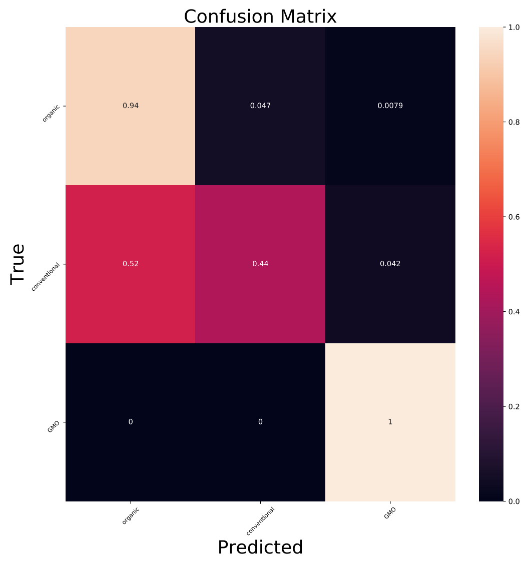
$$\text{classification} = \begin{cases} 1, & \text{if target - interval} \leqslant \text{prediction} \leqslant \text{target + interval} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

4.4.3 Conclusion

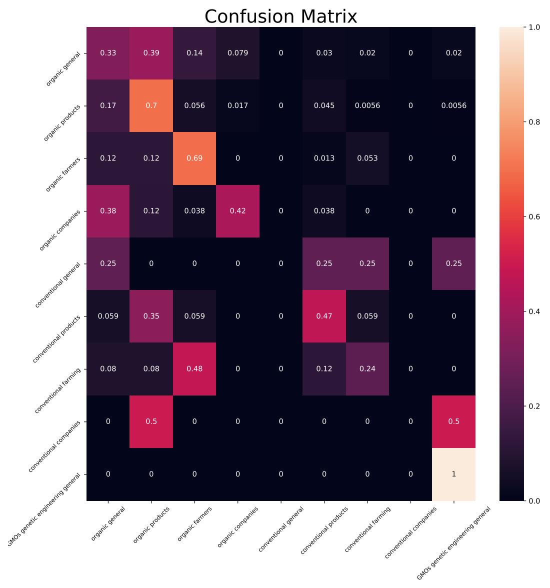
Using the model trained on perception uncertainty targets, obtained through semantic similarities, we can show slight correlation between model predictions and ground truth evaluation targets. However, the correlation is not high enough to definitively prove our novel idea.

Appendix

A Confusion Matrices

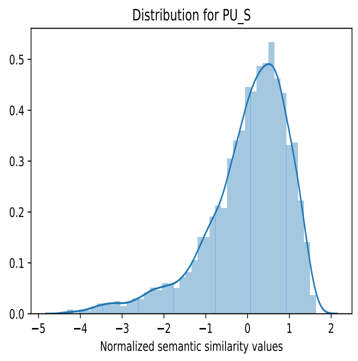


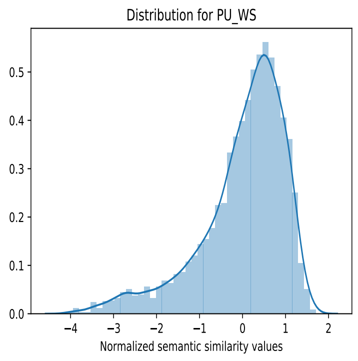
(a) Coarse Entity

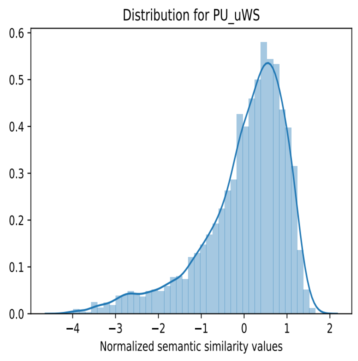


(b) Fine Entity

B Training Targets Distribution







(a) Semantic similarity with class centroid **(b)** Sum of weighted semantic similarities **(c)** Sum of unweighted semantic similarities

C Perception Uncertainty Model Architecture

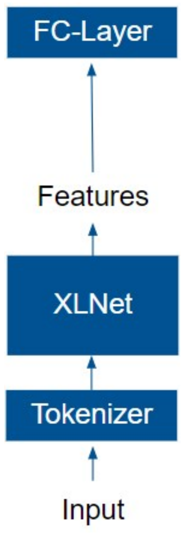
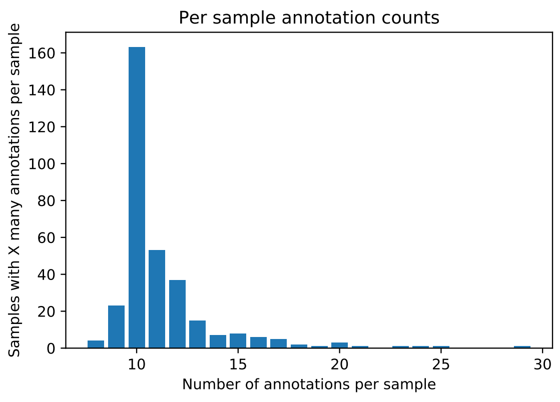
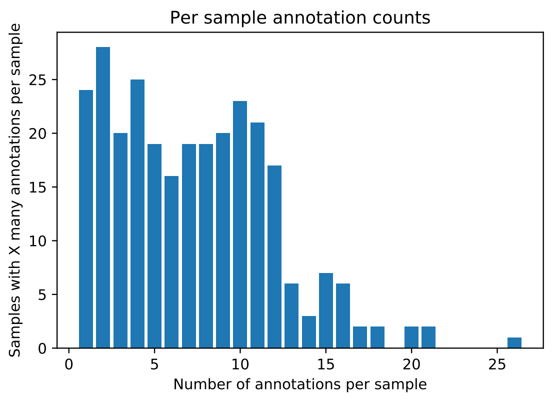


Figure C.1: Model architecture defined for perception uncertainty prediction using XLNet.

D Evaluation Dataset Annotation Count Distribution



(a) Annotation count per sample for all samples



(b) Annotation count per sample for domain relevant samples

E Normalized Information Entropy Targets

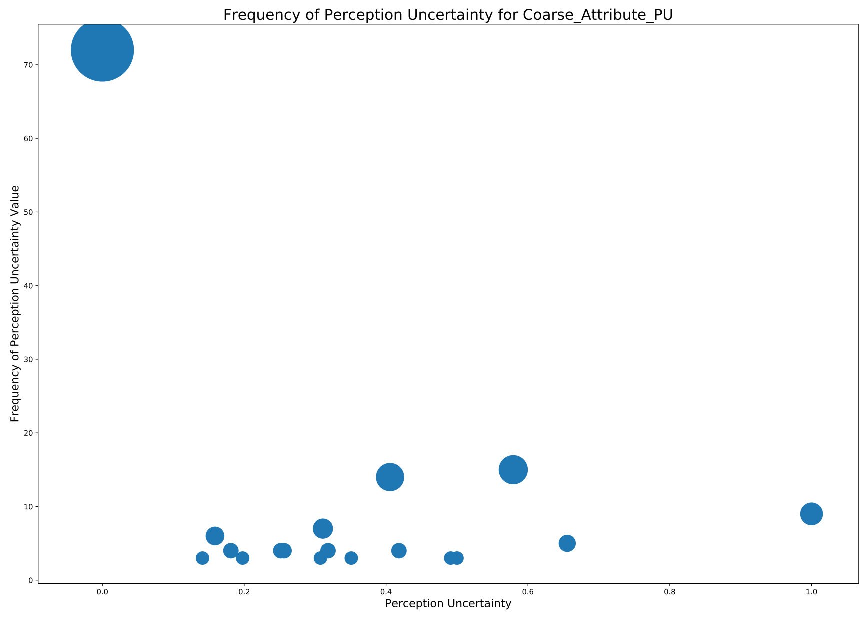


Figure E.1: Distribution of NIE target values for coarse attribute.

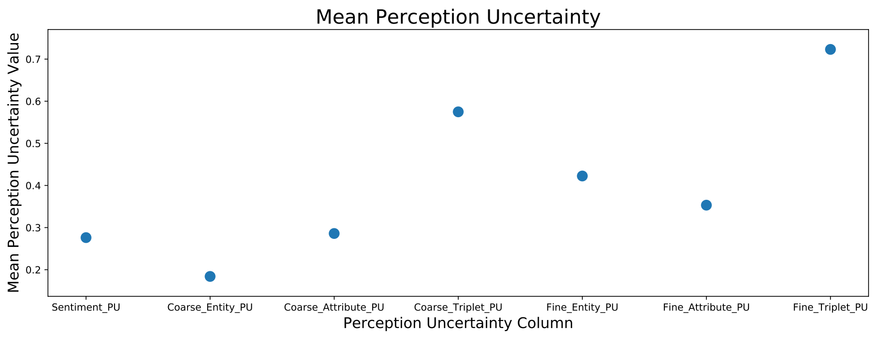


Figure E.2: Mean perception uncertainty values for different ABSA prediction entities.

References

[1] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[4] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, 2015.

[5] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder, 2018.

[6] Mohit Iyyer, Varun Manjunatha, Jordan L. Boyd-Graber, and Hal Daumé. Deep unordered composition rivals syntactic methods for text classification. In *ACL*, 2015.

[7] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2017.

[8] Gerhard Hagerer, Hannah Danner, Farrukh Mushtaq, Phuong Mai Mai, and Georg Groh. Garbage in, garbage seperated: Naive many-rater annotations for aspect-based sentiment analysis. 2019.

[9] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. pages 577–584, 01 2001.