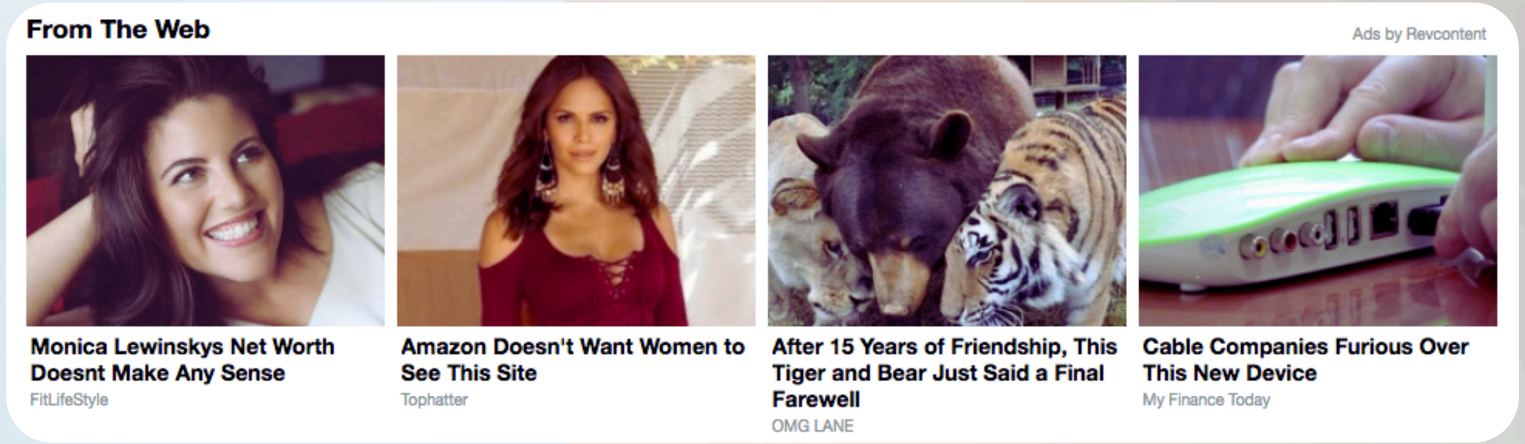
Clickbait Spoiler Detection and Generation



Poluri Rishi Manoj (CS21BTECH11045)
Kota Vignan (CS21BTECH11029)
- Group 17

Problem Statement

- Clickbaits are texts containing sensationalized terms that aim at raising the curiosity of the readers, and lure them to click on a hyperlink. These are common in news headlines, social media posts etc.
- These usually fall short of readers' expectations, wasting their time and causing disappointment.

Due to the deceptive and misleading nature of the content often presented, clickbait is generally considered harmful. Thus addressing the clickbait issue is a crucial concern.

Solution:

- The generation of spoilers, i.e. short texts that neutralize clickbait by providing information that satisfies the curiosity induced by it.

Work Done so far ...

- Implementing the approach given in the paper
  - This involves two steps: 1) finetuning a model for this particular task of spoiler detection, 2) post-hoc ensembling for outputs generated on same model for different seeds
  - Finetuning - used T5-base with huggingface
  - Post-hoc - run T5-base for different seeds (43, 45, 46, 47, 48) and take the best output on each candidate using edit-distance.
- Finalising an approach for project
  - Referred different research papers
  - Discussed ideas and fixed on an approach
- Few initial trials on different models for specific tasks like spoiler type classification
  - Approach involves spoiler type classification
  - Worked out on few different models to find that can best fit to the task

Work Done so far ...

- Implementing classification model
  - Used ‘roberta’ model and finetuned it for classification of spoiler types
  - Each spoiler type is considered as label
- Finetuning model for spoiler generation with spoiler type
  - Used T5-Base and the input now contains spoiler type information as well
  - Finetuned it using training and validation data
  - While testing, classifier predictions are used in the input
- Posthoc ensembling
  - Trained the same model with different seeds and took ensemble of the test predictions
  - Edit distance criteria is used

Results for Baseline

Single

```
BERTScore: {'precision': 0.8847038745880127, 'recall': 0.8751260042190552, 'f1': 0.8792693018913269}
BLEU-4 Score: 0.25643600757035584
TestLoss: 0.22385539564490317
```

Ensemble

```
TestLoss: 0.2194286795258522
BERTScore: {'precision': 0.8899759650230408, 'recall': 0.8760814070701599, 'f1': 0.8823524713516235}
BLEU-4 Score: 0.3095721068976759
```

```
Prediction: Farrah Abraham
Reference: Farrah Abraham
```

```
Prediction: Kim Kardashian Checking Her Makeup
Reference: Kim Kardashian Checking Her Makeup
```

```
Prediction: 1. Your relationship difficulties 2. Your detailed financial state -- or financial problems 3. Your legal woes
Reference: 1. Your relationship difficulties 2. Your detailed financial state -- or financial problems 3. Your legal woes
```

Comments on Results

- Implementing single model (T5-base) gave:
  - BERTScore - ~0.879
  - Bleu - ~0.256
- Implementing ensemble gave:
  - BERTScore - ~0.882
  - Bleu - ~0.309
- Using ensemble gave better results than using a single model, which is better for spoiler generation.
- Observing predictions of the model, the predictions doesn’t generate better spoilers for passage and multipart.

Our Approach

- Spoiler Type Classification
  - Classify the appropriate spoiler type (phrase, passage, or multi-part)
  - Approach: Develop a classifier model that determines which spoiler type best suits the input
- Spoiler Generation
  - After obtaining the spoiler type, include it as part of the prompt or input sequence for each sample and finetune a model to generate spoilers
- Ensembling
  - Generate outputs for a model with different seeds and taking ensemble from them which is Posthoc ensembling

Code Presentation

Results

Description	Num_epochs	Test Loss	Bleu	Bert	Meteor
Single	3	0.20288528645038606	0.3070390288132313	0.8830023407936096	0.2867812215923769
Ensemble	3	0.20104616290330887	0.3099923793616025	0.8842452168464661	0.3069103580464674 4

Classification:

Validation Accuracy: 0.7525

Test Accuracy: 0.746

Link for detailed evaluation scores - https://docs.google.com/document/d/1PePE-S-a48Ek7jsCzZZRoCGDi38LdxB9slqzuAawkKc/edit?usp=sharing

Prediction Examples

```
Prediction: My Activity page
Reference: My Activity page
```

```
Prediction: What’s the one thing you want to do before you die?
Reference: What’s the one thing you want to do before you die?
```

```
Prediction: 1. Clean out your wallet 2. Prioritize your purchases 3. Optimize your credit cards
Reference: 1. Clean out your wallet. 2. Prioritize your purchases. 3. Optimize your credit cards
```

```
Prediction: I am the least qualified person to comment on anyone playing the role of Batman since I so
Reference: "I am the least qualified person to comment on anyone playing the role of Batman since I so terribly destroyed the part," Cloone
y said
```

Comments on Results

- Classification model gave:
  - Test Accuracy - 0.746
- Implementing single model (T5-base) gave:
  - BERTScore - ~0.883
  - Bleu - ~0.307
  - Meteor - ~0.290
- Implementing ensemble gave:
  - BERTScore - ~0.884
  - Bleu - ~0.310
  - Meteor - ~0.310
- Incorporating spoiler type information in the input gave better results
- Ensembling with classification showed a slight improvement in performance
- Observing predictions of the model, the predictions with spoiler type are somewhat good
- The problem is that, the generated predictions doesn’t complete - this is because of the input max_seq_length is set low to avoid memory errors

Challenges Faced

- Trying with large models gave CUDA out of memory errors
- Lack of GPU runtime resources
- Took a bit more time than expected for referring to research papers for a new approach
- Trying to increase input seq length while tokenizing, memory was not enough and it effected the predictions
- Meteor score imports didn’t work for some reason when included with others and it required separate execution file - each time outputs of model have to be saved and given to this execution file
- Trying different combinations like varying learning rate, number of epochs, batch size, max_new_tokens, spoiler type position in the input, seeds for ensemble, etc took more time

Work Division

Rishi Manoj
CS21BTECH11045

- Trying out few models for finetuning
- Implementing and evaluating on the model that gave good results
- Referring to research papers and resources for a different approach
- Involving in discussion
- Trying out combinations of parameters for better output

Vignan
CS21BTECH11029

- Implementing post-hoc ensembling
- Evaluating the model and getting results
- Referring to resources for ideas
- Involving in discussion
- Trying ensembling with different models

Further Improvements

- Spoiler Type Classification:
  - Using a better model with different parameters
  - Condense information from long articles for model input.
  - Improves model performance by focusing on relevant information.
- Spoiler Generation:
  - Using large models like - T5-large, Flan-T5-large, etc with more GPU access
  - Increasing max_length of inputs in tokenization
  - Using appropriate max_new_tokens while generating model predictions
- Ensembling:
  - Combination of different models can be used
  - Different ensembling approach can be used

Thank You