Making the Best Use of Review Summary for Sentiment Analysis

IE643 Course Project

DeepMind2.0

Swapnoneel Kayal 200100154
Yash Choudhary 200100173

October 16, 2022

Outline

1 Main Problem Addressed

2 Details of the Problem

3 Experimental Characteristics

4 Work done by Team

5 Modifications Proposed

6 References

Sentiment Analysis

Sentiment Analysis

A fundamental task in **natural language processing**, which predicts the **subjectivity** and **polarity** of a given text.

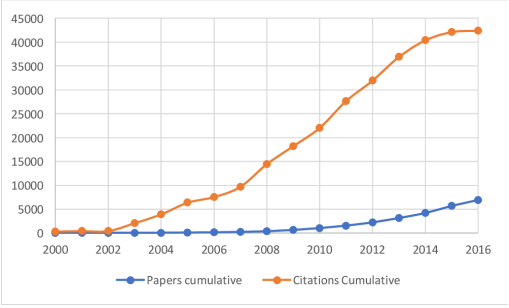
So far, people have trained several models that **only take review** for sentiment classification. However, there are websites which allow the user to give a **summary** in addition to the review. Hence, summaries can be used as **additional training signals**

Under such a scenario, does this ensure us a **better** sentiment classification?

What kind of an architecture can we use to **make best use of both review and summary information** for this purpose?

Background I

The very first instance of sentiment analysis was seen during a **public opinion analysis** at the start of the 20th century.
In the 1990's, **text subjectivity analysis** was performed by the computational linguistics community.
However, the outbreak of computer-based sentiment analysis only occurred with the **availability of subjective texts on the Web**.
Thereafter, 99% of the papers have been published after 2004. [6]



In recent times, people have explored sentiment analysis on even social media texts from Twitter and Facebook.

Background II

Early researchers worked on document-level models and leveraged **machine learning models** like support vector machine, naive Bayes, max entropy, logistic regression and random forest to classify the documents as either positive or negative.
DNN → CNN → RNN → LSTM & GRU → Bi-LSTM
Currently, researchers are working on → **Bi-LSTM with attention**

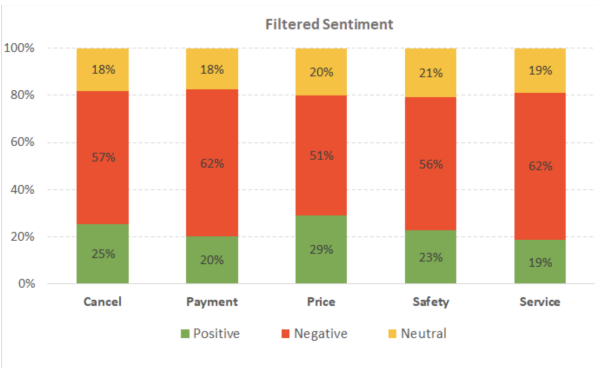
Paper	**Methodology Used**
Chen et al. (2016) [3]	Adopted two GRUs
Xiong et al. (2017) [4]	Made use of co-attention
Zhou et al. (2018) [5]	Adopted both self-attention and cross-attention modules

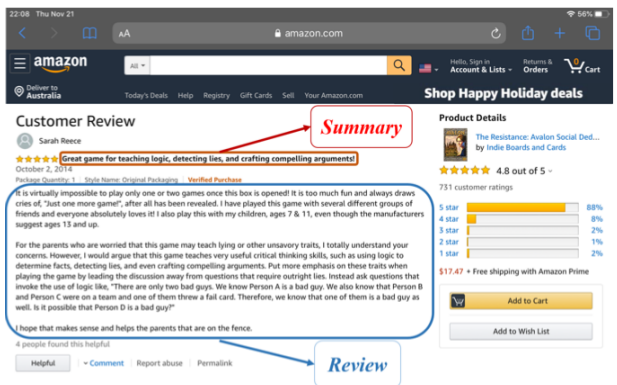
Motivation

Applications of sentiment analysis

They can be applied to any industry, from finance and retail to hospitality and technology.

Intent analysis involves analyzing the **user's intention** behind a message and identifying whether it relates an opinion, news, marketing, complaint, suggestion, appreciation or query.
We present the intent analysis of Facebook comments [7] and a review for *Avalon* from Amazon (**VoC**):



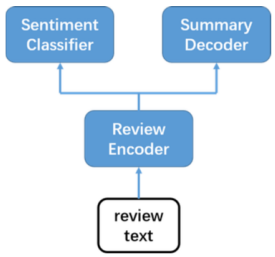


Past Work I

User-written summaries are highly indicative of the user sentiment.

A Hierarchical End-to-End Model for Jointly Improving Text Summarization and Sentiment Classification

Shuming Ma[1], Xu Sun[1], Junyang Lin[2], Xuancheng Ren[1]
[1]MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[2]School of Foreign Languages, Peking University
{shumingma, xusun, linjunyang, renxc}@pku.edu.cn



Proposed a multi-view attention model for joint summarization and sentiment classification

Some of the salient features of this paper [1] :

- Used a **multi-task** model
- Was **only** trained on review information
- Output $\rightarrow$ sentiment and summary

They were able to integrate both the summary and review features into the review encoder using back-propagation

Past Work II

A Self-Attentive Hierarchical Model for Jointly Improving Text Summarization and Sentiment Classification

Hongli Wang WANGHLI8@MAIL2.SYSU.EDU.CN
Jiangtao Ren ISSRJT@MAIL.SYSU.EDU.CN
Schol of Data and Computer Science, Sun Yat-sen University, Guangdong, P.R.China 510006

Some of the salient features of this paper [2] :

- Also exploited the **multi-task** model
- Improved the model of *Ma et al.* [1] by using **additional self-attention** layer on the generated text.
- Model generated more relevant summary which lead to a more accurate summary-aware sentiment prediction

Caveat : Correlation between reviews and summaries can be subtle

- Summary does not convey the sentiment contained in review
- Summary contains explicit sentiment, but the review does not

Task

Problem Formulation

- Input $\rightarrow (X^w, X^s)$ where $X^w = x_1^w, x_2^w, ..., x_n^w$ is a review and $X^s = x_1^s, x_2^s, ..., x_n^s$ is the corresponding summary
- Task $\rightarrow$ Predict sentiment label, y $\in$ [1,5] where 1 $\rightarrow$ most negative sentiment & 5 1 $\rightarrow$ most positive sentiment
- n $\rightarrow$ #words in review & m $\rightarrow$ #words in summary

Questions answered empirically through this research

- What are the roles of and the correlation between a review and its summary for predicting the user rating?
- How to better leverage information from both the review and the summary for effective sentiment classification?

Method

Sequence Encoding

Used Bi-LSTM as the sequence encoder for all experiments.

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i}]$$
$$\mathbf{H} = \{\mathbf{h}_1, ..., \mathbf{h}_m\}$$

Baselines

- Separate Encoding - Two BiLSTMs are adopted to separately encode reviews and summaries
  - Average-pooling baseline
  - Self-attention baseline
- Symmetric Joint Encoding - On top of the sequence encoder, they separately adopted several mechanisms
  - Pooling
  - Self-attention
  - Hard-attention
  - Co-attention

Review-centric Joint Encoding

- **Changed only the review encoder** over the baseline
- The review encoder has a set of **stacked layers**, each consisting of a sequence encoding sublayer and an attention inference sublayer
- The **sequence encoding sublayers** takes the same BiLSTM structure as the summary encoder, but with different model parameters
- The **attention inference sublayer** integrates summary information into the review representation

$$
\begin{aligned}
\boldsymbol{\alpha} &= \mathrm{softmax}(\frac{\mathbf{H}^w\mathbf{W}_i^Q(\mathbf{h}^s\mathbf{W}_i^K)^\intercal}{\sqrt{d_h/k}}) \\
\mathbf{head}_i &= \hat{\mathbf{A}}(\hat{\mathbf{H}}^s)^\intercal\mathbf{W}_i^V \\
\mathbf{H}^{w,s} &= \mathrm{concat}(\mathbf{head}_1, \ldots, \mathbf{head}_k),
\end{aligned}
$$

where superscripts w and s represent review and summary, respectively. $\hat{\mathbf{A}}$ and $\hat{\mathbf{H}}^s$ are the unsqueezed matrices of $\boldsymbol{\alpha}$ and $\mathbf{h}^s$. $\mathbf{W}_i^Q \in \mathbb{R}^{d_h \times \frac{d_h}{k}}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_h \times \frac{d_h}{k}}$ and $\mathbf{W}_i^V \in \mathbb{R}^{d_h \times \frac{d_h}{k}}$ are model parameters. Q, K and V represent *Query*, *Key* and *Value*, respectively. k is the number of parallel heads and $i \in [1, k]$ indicates which head is being processed.

$$\mathbf{H} = \mathrm{LayerNorm}(\mathbf{H}^w + \mathbf{H}^{w,s})$$

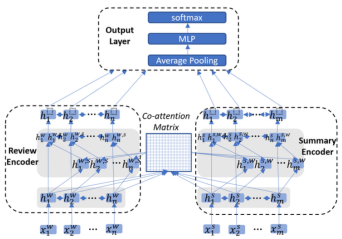
Adopted a residual connection around each attention inference layer

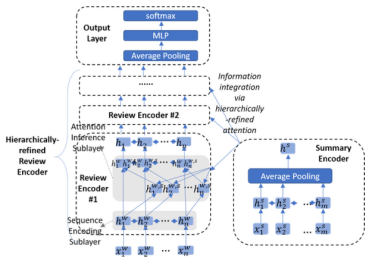
$$H = LayerNorm(H^w + H^{w,s})$$

H is then fed to the subsequent sequence encoding layer as input, if any.

Output Layer

Architectures of co-attention model & proposed review-centric model :





Global average pooling is applied on **H**, followed by a classifier layer:

$$\begin{aligned} \mathbf{h}^{avg} &= \text{avg-pooling}(\mathbf{h}_1, ..., \mathbf{h}_n) \\ \mathbf{p} &= \text{softmax}(\mathbf{W}\mathbf{h}^{avg} + \mathbf{b}) \\ \hat{y} &= \text{argmax}\,\mathbf{p}, \end{aligned}$$

where $\hat{y}$ is the predicted sentiment label; **W** and **b** are parameters to be learned during **training**.

Models were trained by using the following **Loss Function** :

$$L = -\sum_{t=1}^{|T|} \log(\mathbf{p}^{[y_t]})$$

where $p^{[yt]}$ denotes the value of the label in **p** that corresponds to y_t.

The Dataset

Dataset Used : SNAP Amazon review datasets [8]

- Stands for **Stanford Network Analysis Project**
- Includes around **34 million Amazon** reviews in different domains. They only **considered three datasets** (Toys & Games, Sports & Outdoors and Movies & TV). Why?

For each dataset, the first **1000** samples are taken as the development set, the next **1000** samples as the test set, and the rest as the training set.

Domain	Size	#Review	#Summary
Toys & Games	168k	99.9	4.4
Sports & Outdoors	296k	87.2	4.2
Movies & TV	1,698k	161.6	4.8

Frameworks and Libraries

- **Computational framework considered** : cudatoolkit==9.0
- **Programming language used** : python==3.6
- **Programming framework used** : pytorch==1.0.1
- **Packages required** : tensorboardX, pickle, nltk, numpy

Competing Models

- Baseline models using review only
- Baseline models using summary only
- Separate Encoder Models
- Joint Encoder Models
- Multi - Task models like **HSSC** [1] & **SAHSSC** [2]

Experimental Settings

- Used **GloVe**. Stands for "Global Vectors". It is an unsupervised learning algorithm for obtaining vector representations for words.
- **LSTM hidden size** $\rightarrow$ 256
- **Optimizer** for all models $\rightarrow$ Adam
  - **Learning rate** $\rightarrow$ $3e{-}4$, e $= 1 \times 10^{-8}$
  - **Momentum parameters** $\rightarrow$ $\beta_1 = 0.9$, $\beta_2 = 0.999$
- **Dropout rate**, α and **Number of attention heads**, M
  - For Toys & Games: $\alpha = 0.5$, **M** $= 1$
  - For Sports & Outdoors: $\alpha = 0.2$, **M** $= 1$
  - For Movies & TV: $\alpha = 0.0$, **M** $= 2$
- Number of layers for the review-centric model $\rightarrow$ **2**

For generating summaries, they separately adopted a pointer-generator network (PG-Net) with coverage mechanism [9] trained on the training set.

Important Results

The review-centric model gave **1.3%** and **0.5%** improvements compared with the best baseline (co-attention) with both gold summary and system-generated summary, respectively.

Model	T & G	S & O	M & T	Average
Multi-task				
HSSC	71.9	73.2	68.9	71.3
SAHSSC	72.5	–	69.2	70.9
Summary Only				
Pooling	73.0	69.5	68.1	70.2
Self-attention	71.9	70.4	68.9	70.4
Review Only				
Pooling	73.3	71.2	71.7	72.1
Self-attention	73.5	71.8	72.3	72.5
Separate Encoder				
Pooling	74.4	73.9	73.8	74.0
Self-attention	75.8	73.1	73.7	74.2
Joint Encoder				
Hard-attention	73.4	72.1	73.9	73.1
Pooling	75.4	73.4	73.2	74.0
Self-attention	75.7	74.3	74.1	74.7
Co-attention	76.1	74.2	74.3	74.9
Review-centric Model	**76.6**	**76.1**	**75.9**	**76.2**

Figure: Results obtained when using gold summary as input.

Model	T & G	S & O	M & T	Average
Separate Encoder				
Pooling	71.8	72.2	72.5	72.2
Self-attention	73.1	72.5	72.6	72.7
Joint Encoder				
Pooling	73.8	72.0	72.0	72.6
Self-attention	73.9	71.6	72.4	72.6
Co-attention	73.8	72.2	72.7	72.9
Review-centric Model	**74.8**	**72.6**	**72.8**	**73.4**

Figure: Results obtained when using system-generated summary as input.

Model	T & G	S & O	M & T	Average
Co-attention (*review*)	75.1	74.8	74.7	74.9
Co-attention (*summary*)	74.1	74.2	73.4	73.9
Co-attention (*concat*)	76.1	74.2	74.3	74.9
Summary-centric	73.8	74.5	74.9	74.4
Review-centric	76.6	76.1	75.9	76.2

Figure: Result comparison among different interacting schemes

Details of Initial Experiment setup

- **Local PC setup** along with **Cuda Environment** for PyTorch and **replication of code** are completed (Version 1.21.1 for PyTorch and 11.3 for CUDA)
- **Dataset was accessed** for Amazon reviews on Toys and Games for a **sample run of the model**.
- The data preprocessing part was successfully completed. It included the following :
  - Tokenizing the dataset
  - Processing word embedding files for the given dataset
  - Saving the tokenized dataset, word to index, and index to vector pickle files
  - Using them while training

Status of Experiments and Results

- Time taken to Data Tokenization $\rightarrow$ 2.3644 seconds
- Time taken to Data embedding $\rightarrow$ 1.4 seconds
- Training of the model with the processed dataset started and the following results were obtained :

	Epoch #1		Epoch #2	
	Accuracy	Loss	Accuracy	Loss
Training Set	0.672	0.0060	0.692	0.0057
Development Set	0.689	0.0055	0.713	0.00521
Test Set	0.696	0.0057	0.728	0.00524

The training didn't show any progress since our setup was running out of GPU memory, hence we needed to find a solution for it.

Modifications

The main modification that we plan on experimenting with is to **change the Bi-LSTM block of the model to a Bi-GRU**.
Some other modifications we felt were worth trying out:-

- Train on a smaller subset of the training data and then consider increasing the subset. Why?
- Change the number of layers present in their proposed model
- Use dropout instead of the global average pooling
- For the generation of summaries, can we replace the PG-net with something more state-of-the-art
- Why just restrict the usage of the summary generation model to cases wherein the user has not provided the summary. We can add the summary generated by the model to the actual summary given by the user. Why?

There were a lot of ideas that we had discussed but we spent time on understanding and selecting the ones that are worth trying

References

[1] : https://arxiv.org/abs/1805.01089

[2] : https://arxiv.org/abs/1805.01089

[3] : https://aclanthology.org/P16-1223/

[4] : https://arxiv.org/abs/1611.01604

[5] : https://arxiv.org/abs/1612.01627

[6] : https://sci-hub.se/https://doi.org/10.1016/j.cosrev.2017.10.002

[7] : https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications

[8] : https://snap.stanford.edu/data/web-Amazon.html

[9] : https://arxiv.org/abs/1704.04368