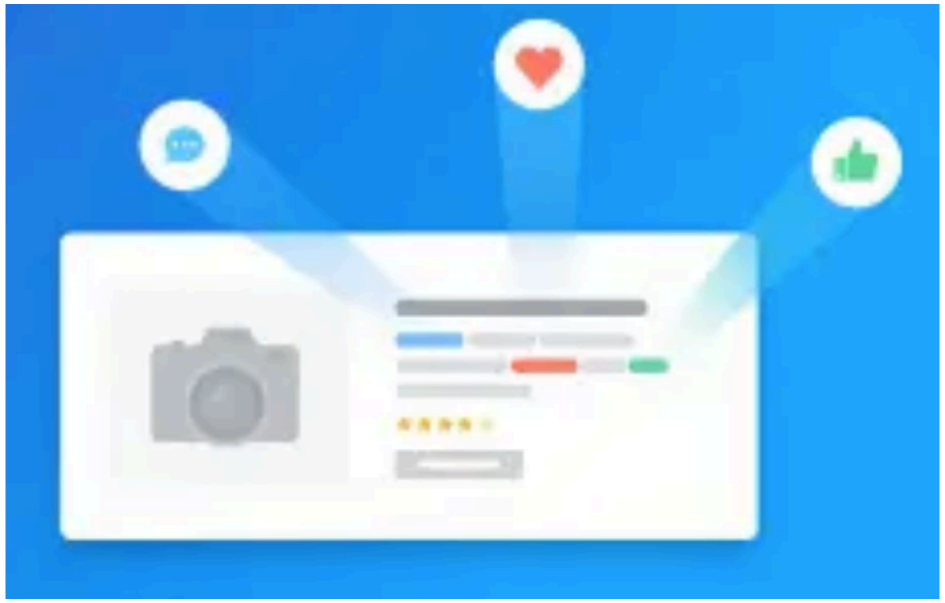
Progressive Self-Supervised Attention Learning for Aspect-Level Sentiment Analysis

Jialong Tang, Ziyao Lu, Jinsong Su, Yubin Ge,
Linfeng Song, Le Sun, Jiebo Luo

Present at "DL-NLP RG" by Azadeh Hashemi
September 26, 2019

Background: Sentiment Analysis

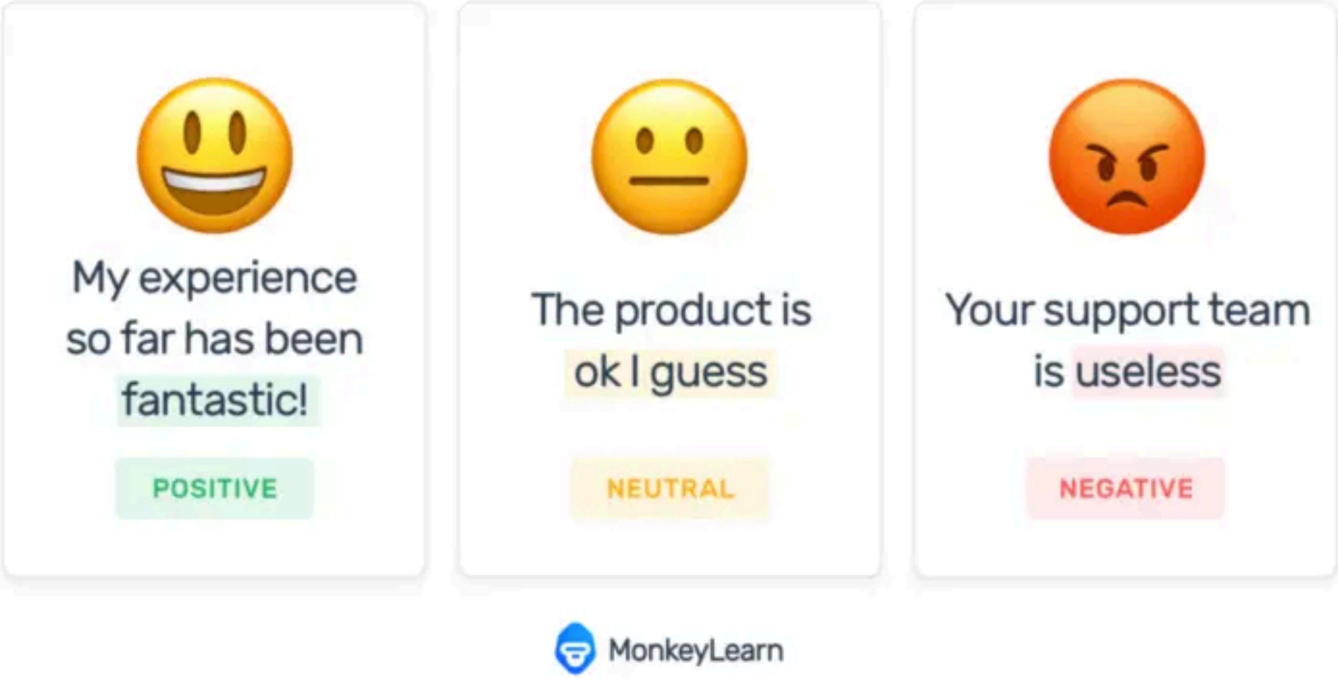
- Sentiment analysis is the automated process that allows machines to identify and extract opinions within text, such as tweets, emails, support tickets, product reviews, survey responses, etc.



https://monkeylearn.com/sentiment-analysis/

Background: Sentiment Analysis (cont.)

- Usually, besides identifying the opinion, Sentiment Analysis systems extract attributes of the expression e.g.:
  - *Polarity*: if the speaker express a *positive* or *negative* opinion,
  - *Subject*: the thing that is being talked about,
  - *Opinion holder*: the person, or entity that expresses the opinion.



https://monkeylearn.com/sentiment-analysis/

Background: Aspect-Based Sentiment Analysis

- Instead of classifying the overall sentiment of a text into positive or negative, aspect-based analysis allows us to associate specific sentiments with different aspects of a product or service.

- Here’s a breakdown of what aspect-based sentiment analysis can extract:

  - Sentiments: positive or negative opinions about a particular aspect.

  - Aspects: the thing or topic that is being talked about.

Introduction: Aspect-level Sentiment Classification (ASC)

- Previous representative models are mostly discriminative classifiers based on manual feature engineering, such as Support Vector Machine.

- Recently, dominant ASC models have evolved into neural network (NN) models which are able to automatically learn the aspect-related semantic representation.

NN-based models equipped with attention mechanism

- Major drawback:
  It is prone to overly focus on a few frequent words with sentiment polarities and little attention is laid upon low-frequency ones.

Type	Sentence	Ans./Pred.
Train	The **[place]** is small and crowded but the service is quick .	Neg / —
Train	The **[place]** is a bit too small for live music .	Neg / —
Train	The service is decent even when this small **[place]** is packed .	Neg / —
Test	At lunch time , the **[place]** is crowded .	Neg / Pos
Test	A small area makes for quiet **[place]** to study alone .	Pos / Neg

Table 1: The example of attention visualization for five sentences, where the first three are training instances and the last two are test ones. The bracketed bolded words are target aspects. Ans./Pred. = ground-truth/predicted sentiment label. Words are highlighted with different degrees according to attention weights.

Contribution

- They propose a novel progressive self-supervised attention learning approach for neural ASC models.
- The method is able to automatically and incrementally mine attention supervision information from a training corpus, which can be exploited to guide the training of attention mechanisms in ASC models.

Notations

- Input sentence: $x = (x_1, x_2, ..., x_N)$
- Given target aspect: $t = (t_1, t_2, ..., t_T)$
- Ground-truth sentiment: y
- Predicted sentiment: $y_p \in \{\text{Positive, Negative, Neutral}\}$
- Aspect embedding matrix: $v(t)$

MN

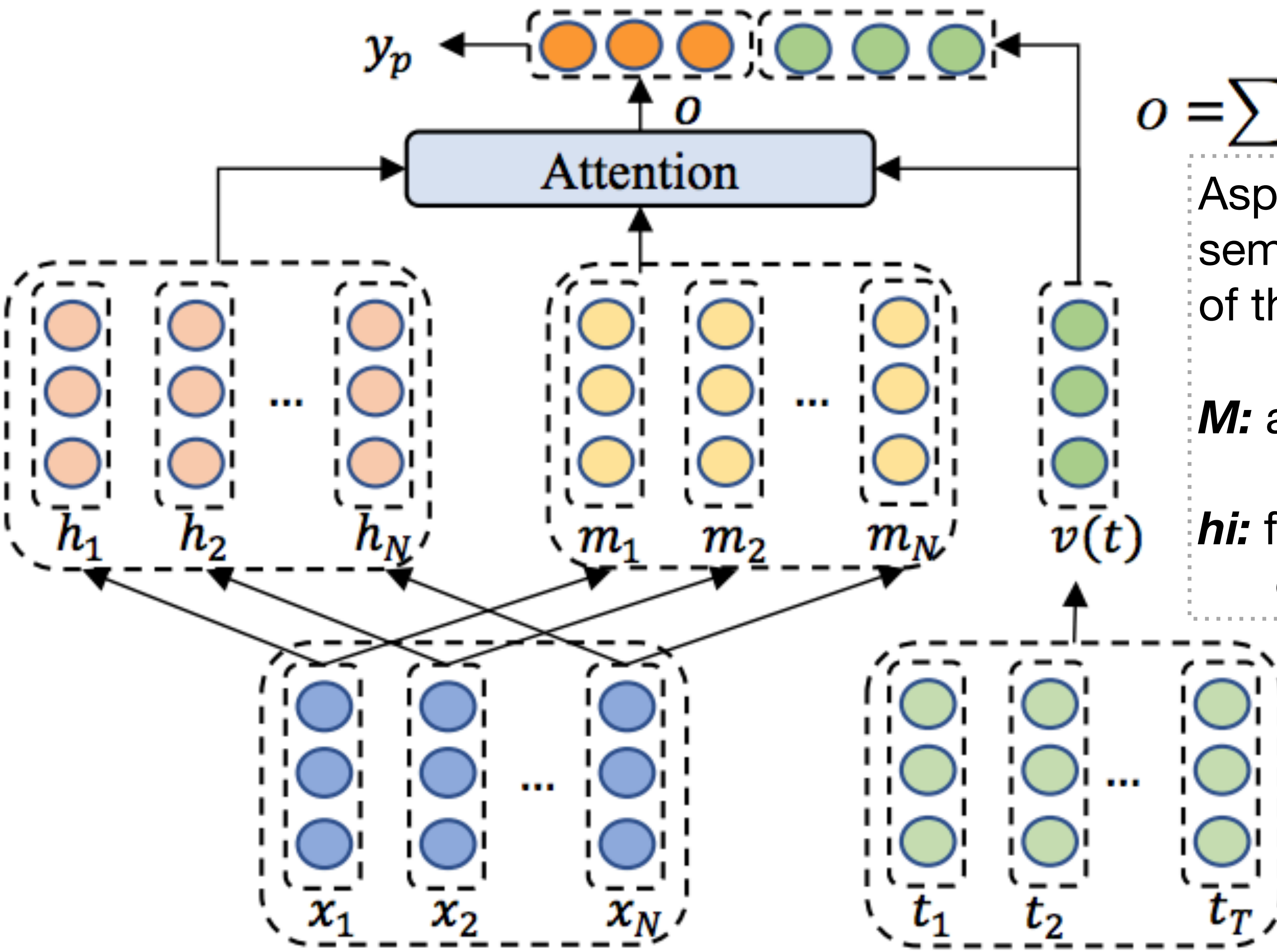


Figure 1: The framework architecture of MN.

$$o = \sum_i softmax(v_t^T M m_i) h_i$$

Aspect-related semantic representation ***o*** of the sentence ***x***

M: attention matrix

hi: final semantic representation of ***xi***

TNet

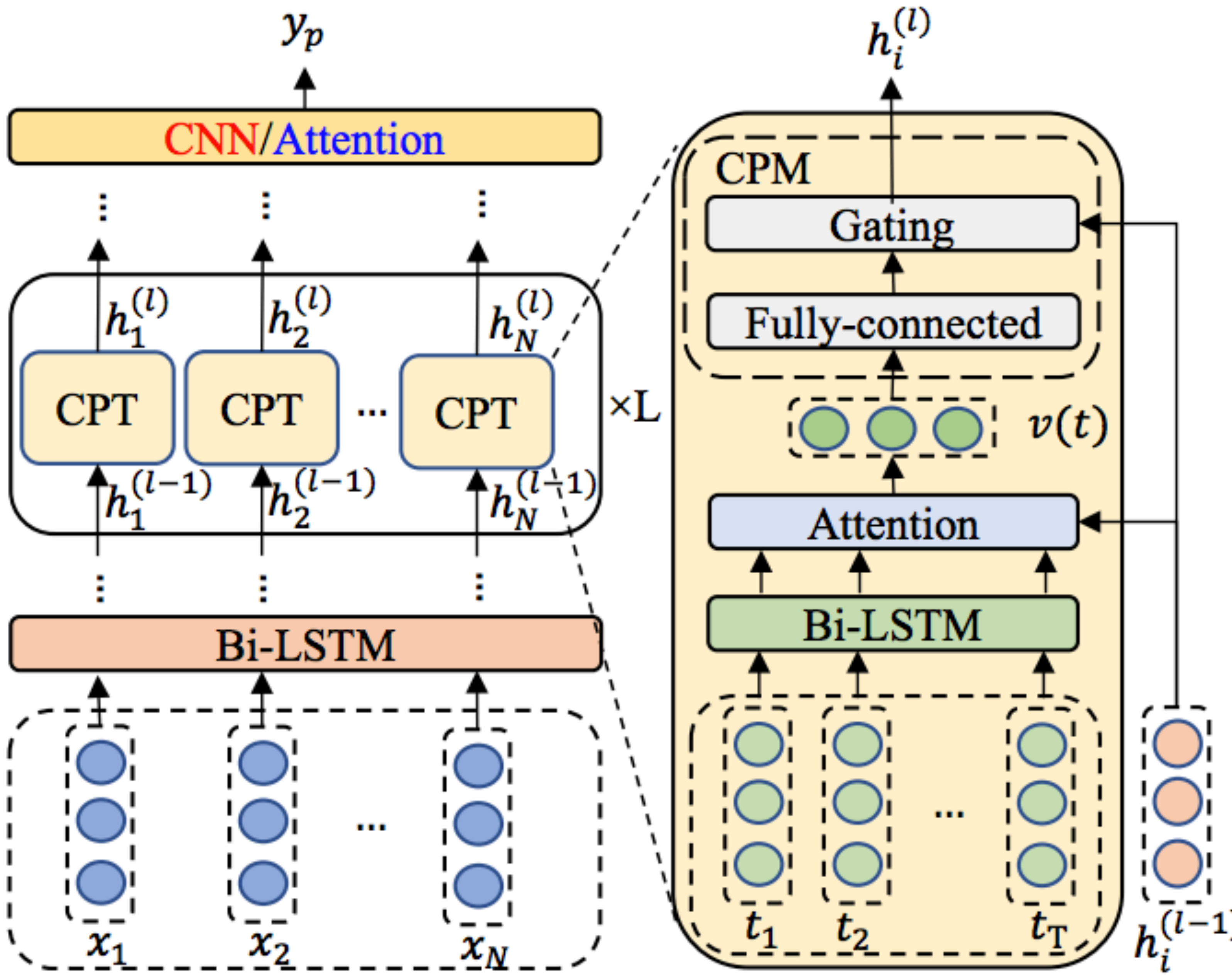


Figure 2: The framework architecture of TNet/TNet-ATT. Note that TNet-ATT is the variant of TNet replacing CNN with an attention mechanism.

Training Objective

- Negative log-likelihood of the gold-truth sentiment tags:

$$J(D;\theta) = -\sum_{(x,t,y)\in D} J(x,t,y;\theta)$$
$$= \sum_{(x,t,y)\in D} d(y) \cdot \log d(x,t;\theta),$$

D is the training corpus, ***d(y)*** is the one-hot vector of ***y***, ***d(x,t;θ)*** is the model-predicted sentiment distribution for the pair ***(x,t)***

Basic Intuition

- Context word with the maximum attention weight ->
- Often the one with strong sentiment polarity ->
- Usually occurs frequently in the training corpus ->
- Thus tends to be overly considered during model training ->
- This simultaneously leads to the insufficient learning of other context words, especially low-frequency ones with sentiment polarities.

Basic Intuition (cont.)

- The importance of each context word on the given aspect mainly depends on its attention weight.
- Thus, the context word with the maximum attention weight has the most important impact on the sentiment prediction of the input sentence.
- Therefore, for a training sentence, if the prediction of ASC model is correct, we believe that it is reasonable to continue focusing on this context word.
- Conversely, the attention weight of this context word should be decreased.

Basic Intuition (cont.)

- One intuitive and feasible method :
  - First shield the influence of this most important context word before reinvestigating effects of remaining context words of the training instance.
  - In that case, other low-frequency context words with sentiment polarities can be discovered according to their attention weights.

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;

θ^{init}: the initial model parameters;

ϵ_α: the entropy threshold of attention weight distribution;

K: the maximum number of training iterations;

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
θ^{init}: the initial model parameters;
ϵ_α: the entropy threshold of attention weight distribution;
K: the maximum number of training iterations;

1: $\theta^{(0)} \leftarrow \textbf{\textit{Train}}(D; \theta^{init})$

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;

θ^{init}: the initial model parameters;

ϵ_α: the entropy threshold of attention weight distribution;

K: the maximum number of training iterations;

$\theta^{(0)} \leftarrow \boldsymbol{Train}(D; \theta^{init})$
for $(x, t, y) \in D$ **do**
    $s_a(x) \leftarrow \emptyset$
    $s_m(x) \leftarrow \emptyset$
end for

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
θ^{init}: the initial model parameters;
ϵ_α: the entropy threshold of attention weight distribution;
K: the maximum number of training iterations;

1: $\theta^{(0)} \leftarrow$ ***Train***$(D; \theta^{init})$
2: **for** $(x, t, y) \in D$ **do**
3: $\quad s_a(x) \leftarrow \emptyset$
4: $\quad s_m(x) \leftarrow \emptyset$
5: **end for**

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
θ^{init}: the initial model parameters;
ϵ_α: the entropy threshold of attention weight distribution;
K: the maximum number of training iterations;

```
θ^(0) ← Train(D; θ^init)
for (x, t, y) ∈ D do
    s_a(x) ← ∅
    s_m(x) ← ∅
end for
for k = 1, 2..., K do
    D^(k) ← ∅
    for (x, t, y) ∈ D do
```

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
θ^{init}: the initial model parameters;
ϵ_α: the entropy threshold of attention weight distribution;
K: the maximum number of training iterations;

```
θ^(0) ← Train(D; θ^init)
for (x, t, y) ∈ D do
    s_a(x) ← ∅
    s_m(x) ← ∅
end for
for k = 1, 2..., K do
    D^(k) ← ∅
    for (x, t, y) ∈ D do
        v(t) ← GenAspectRep(t, θ^(k−1))
```

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
θ^{init}: the initial model parameters;
ϵ_α: the entropy threshold of attention weight distribution;
K: the maximum number of training iterations;

$\theta^{(0)} \leftarrow$ ***Train***$(D; \theta^{init})$
for $(x, t, y) \in D$ **do**
  $s_a(x) \leftarrow \emptyset$
  $s_m(x) \leftarrow \emptyset$
end for

for $k = 1, 2..., K$ **do**
  $D^{(k)} \leftarrow \emptyset$
  for $(x, t, y) \in D$ **do**
    $v(t) \leftarrow$ ***GenAspectRep***$(t, \theta^{(k-1)})$
    $x' \leftarrow$ ***MaskWord***$(x, s_a(x), s_m(x))$

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
θ^{init}: the initial model parameters;
ϵ_α: the entropy threshold of attention weight distribution;
K: the maximum number of training iterations;

$\theta^{(0)} \leftarrow$ ***Train***$(D; \theta^{init})$
for $(x, t, y) \in D$ **do**
  $s_a(x) \leftarrow \emptyset$
  $s_m(x) \leftarrow \emptyset$
end for
for $k = 1, 2..., K$ **do**
  $D^{(k)} \leftarrow \emptyset$
  for $(x, t, y) \in D$ **do**
    $v(t) \leftarrow$ ***GenAspectRep***$(t, \theta^{(k-1)})$
    $x' \leftarrow$ ***MaskWord***$(x, s_a(x), s_m(x))$
    $h(x') \leftarrow$ ***GenWordRep***$(x', v(t), \theta^{(k-1)})$

$$\boxed{h(x')=\{h(x'_i)\}_{i=1}^{N}}$$

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
θ^{init}: the initial model parameters;
ϵ_α: the entropy threshold of attention weight distribution;
K: the maximum number of training iterations;

```
1: θ^(0) ← Train(D; θ^init)
2: for (x, t, y) ∈ D do
3:     s_a(x) ← ∅
4:     s_m(x) ← ∅
5: end for
6: for k = 1, 2..., K do
7:     D^(k) ← ∅
8:     for (x, t, y) ∈ D do
9:         v(t) ← GenAspectRep(t, θ^(k−1))
10:        x′ ← MaskWord(x, s_a(x), s_m(x))
11:        h(x′) ← GenWordRep(x′, v(t), θ^(k−1))
12:        y_p, α(x′) ← SentiPred(h(x′), v(t), θ^(k−1))
```

(**Line 12**), where the word-level attention weight distribution $\alpha(x')=\{\alpha(x'_1), \alpha(x'_2), ..., \alpha(x'_N)\}$ subjecting to $\sum_{i=1}^{N} \alpha(x'_i) = 1$ is induced.

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
θ^{init}: the initial model parameters;
ϵ_α: the entropy threshold of attention weight distribution;
K: the maximum number of training iterations;

$\theta^{(0)} \leftarrow$ ***Train***$(D; \theta^{init})$
for $(x, t, y) \in D$ **do**
    $s_a(x) \leftarrow \emptyset$
    $s_m(x) \leftarrow \emptyset$
end for
for $k = 1, 2..., K$ **do**
    $D^{(k)} \leftarrow \emptyset$
    for $(x, t, y) \in D$ **do**
        $v(t) \leftarrow$ ***GenAspectRep***$(t, \theta^{(k-1)})$
        $x' \leftarrow$ ***MaskWord***$(x, s_a(x), s_m(x))$
        $h(x') \leftarrow$ ***GenWordRep***$(x', v(t), \theta^{(k-1)})$
        $y_p, \alpha(x') \leftarrow$ ***SentiPred***$(h(x'), v(t), \theta^{(k-1)})$
        $E(\alpha(x')) \leftarrow$ ***CalcEntropy***$(\alpha(x'))$

$$\boxed{E(\alpha(x')) = -\sum_{i=1}^{N} \alpha(x'_i) \log(\alpha(x'_i))}$$

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
θ^{init}: the initial model parameters;
ϵ_α: the entropy threshold of attention weight distribution;
K: the maximum number of training iterations;

```
θ^(0) ← Train(D; θ^init)
for (x, t, y) ∈ D do
    s_a(x) ← ∅
    s_m(x) ← ∅
end for
for k = 1, 2..., K do
    D^(k) ← ∅
    for (x, t, y) ∈ D do
        v(t) ← GenAspectRep(t, θ^(k−1))
        x' ← MaskWord(x, s_a(x), s_m(x))
        h(x') ← GenWordRep(x', v(t), θ^(k−1))
        y_p, α(x') ← SentiPred(h(x'), v(t), θ^(k−1))
        E(α(x')) ← CalcEntropy(α(x'))
        if E(α(x')) < ε_α then
            m ← argmax_{1≤i≤N} α(x'_i)
            if y_p == y then
                s_a(x) ← s_a(x) ∪ {x'_m}
            else
                s_m(x) ← s_m(x) ∪ {x'_m}
            end if
        end if
```

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
θ^{init}: the initial model parameters;
ϵ_α: the entropy threshold of attention weight distribution;
K: the maximum number of training iterations;

$\theta^{(0)} \leftarrow \textbf{\textit{Train}}(D; \theta^{init})$
for $(x, t, y) \in D$ **do**
  $s_a(x) \leftarrow \emptyset$
  $s_m(x) \leftarrow \emptyset$
end for
for $k = 1, 2..., K$ **do**
  $D^{(k)} \leftarrow \emptyset$
  for $(x, t, y) \in D$ **do**
    $v(t) \leftarrow \textbf{\textit{GenAspectRep}}(t, \theta^{(k-1)})$
    $x' \leftarrow \textbf{\textit{MaskWord}}(x, s_a(x), s_m(x))$
    $h(x') \leftarrow \textbf{\textit{GenWordRep}}(x', v(t), \theta^{(k-1)})$
    $y_p, \alpha(x') \leftarrow \textbf{\textit{SentiPred}}(h(x'), v(t), \theta^{(k-1)})$
    $E(\alpha(x')) \leftarrow \textbf{\textit{CalcEntropy}}(\alpha(x'))$
    if $E(\alpha(x')) < \epsilon_\alpha$ **then**
      $m \leftarrow argmax_{1 \le i \le N}\ \alpha(x'_i)$
      if $y_p == y$ **then**
        $s_a(x) \leftarrow s_a(x) \cup \{x'_m\}$
      else
        $s_m(x) \leftarrow s_m(x) \cup \{x'_m\}$
      end if
    end if
    $D^{(k)} \leftarrow D^{(k)} \cup (x', t, y)$
  end for
  $\theta^{(k)} \leftarrow \textbf{\textit{Train}}(D^{(k)}; \theta^{(k-1)})$
end for

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
θ^{init}: the initial model parameters;
ϵ_α: the entropy threshold of attention weight distribution;
K: the maximum number of training iterations;

$\theta^{(0)} \leftarrow$ ***Train***$(D; \theta^{init})$
for $(x, t, y) \in D$ **do**
    $s_a(x) \leftarrow \emptyset$
    $s_m(x) \leftarrow \emptyset$
end for
for $k = 1, 2..., K$ **do**
    $D^{(k)} \leftarrow \emptyset$
    for $(x, t, y) \in D$ **do**
        $v(t) \leftarrow$ ***GenAspectRep***$(t, \theta^{(k-1)})$
        $x' \leftarrow$ ***MaskWord***$(x, s_a(x), s_m(x))$
        $h(x') \leftarrow$ ***GenWordRep***$(x', v(t), \theta^{(k-1)})$
        $y_p, \alpha(x') \leftarrow$ ***SentiPred***$(h(x'), v(t), \theta^{(k-1)})$
        $E(\alpha(x')) \leftarrow$ ***CalcEntropy***$(\alpha(x'))$
        if $E(\alpha(x')) < \epsilon_\alpha$ **then**
            $m \leftarrow argmax_{1 \leq i \leq N}\ \alpha(x'_i)$
            if $y_p == y$ **then**
                $s_a(x) \leftarrow s_a(x) \cup \{x'_m\}$
            else
                $s_m(x) \leftarrow s_m(x) \cup \{x'_m\}$
            end if
        end if
        $D^{(k)} \leftarrow D^{(k)} \cup (x', t, y)$
    end for
    $\theta^{(k)} \leftarrow$ ***Train***$(D^{(k)}; \theta^{(k-1)})$
end for

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
θ^{init}: the initial model parameters;
ϵ_α: the entropy threshold of attention weight distribution;
K: the maximum number of training iterations;

$\theta^{(0)} \leftarrow$ ***Train***$(D; \theta^{init})$
for $(x, t, y) \in D$ **do**
    $s_a(x) \leftarrow \emptyset$
    $s_m(x) \leftarrow \emptyset$
end for
for $k = 1, 2..., K$ **do**
    $D^{(k)} \leftarrow \emptyset$
    for $(x, t, y) \in D$ **do**
        $v(t) \leftarrow$ ***GenAspectRep***$(t, \theta^{(k-1)})$
        $x' \leftarrow$ ***MaskWord***$(x, s_a(x), s_m(x))$
        $h(x') \leftarrow$ ***GenWordRep***$(x', v(t), \theta^{(k-1)})$
        $y_p, \alpha(x') \leftarrow$ ***SentiPred***$(h(x'), v(t), \theta^{(k-1)})$
        $E(\alpha(x')) \leftarrow$ ***CalcEntropy***$(\alpha(x'))$
        if $E(\alpha(x')) < \epsilon_\alpha$ **then**
            $m \leftarrow argmax_{1 \leq i \leq N}\ \alpha(x'_i)$
            if $y_p == y$ **then**
                $s_a(x) \leftarrow s_a(x) \cup \{x'_m\}$
            else
                $s_m(x) \leftarrow s_m(x) \cup \{x'_m\}$
            end if
        end if
        $D^{(k)} \leftarrow D^{(k)} \cup (x', t, y)$
    end for
    $\theta^{(k)} \leftarrow$ ***Train***$(D^{(k)}; \theta^{(k-1)})$
end for
for $(x, t, y) \in D$ **do**
    $D_s \leftarrow D_s \cup (x, t, y, s_a(x), s_m(x))$
end for
$\theta \leftarrow$ ***Train***(D_s)
Return: θ;

Example

Iter	Sentence	Ans./Pred.	$E(\alpha(x'))$	x'_m
1	The **[place]** is small and crowded but the service is quick .	**Neg / Neg**	**2.38**	***small***
2	The **[place]** is $\langle mask \rangle$ and crowded but the service is quick .	**Neg / Neg**	**2.59**	***crowded***
3	The **[place]** is $\langle mask \rangle$ and $\langle mask \rangle$ but the service is quick .	**Neg / Pos**	**2.66**	***quick***
4	The **[place]** is $\langle mask \rangle$ and $\langle mask \rangle$ but the service is $\langle mask \rangle$.	**Neg / Neg**	**3.07**	—

Table 2: The example of mining influential context words from the first training sentence in Table 1. $E(\alpha(x'))$ denotes the entropy of the attention weight distribution $\alpha(x')$, ϵ_α is entropy threshold set as 3.0, and x'_m indicates the context word with the maximum attention weight. Note that all extracted words are replaced with "$\langle mask \rangle$"

Model training with attention supervision information

- Soft attention Regularizer:

$$\boxed{\triangle(\alpha(s_a(x) \cup s_m(x)), \hat{\alpha}(s_a(x) \cup s_m(x)); \theta)}$$

  - $\alpha(*)$: the model-induced
  - $\hat{\alpha}(*)$: the expected attention weight distributions of words in $s_a(x) \cup s_m(x)$
  - $\triangle(\alpha(*), \hat{\alpha}(*); \theta)$: Euclidean Distance style loss that penalize the disagreement

Objective Function

$$J(D;\theta) = - \sum_{(x,t,y)\in D} J(x,t,y;\theta)$$
$$= \sum_{(x,t,y)\in D} d(y) \cdot \mathbf{log} d(x,t;\theta),$$

$$J_s(D_s;\theta) = - \sum_{(x,t,y)\in D_s} \{J(x,t,y;\theta) +$$
$$\gamma\Delta(\alpha(s_a(x) \cup s_m(x)), \hat{\alpha}(s_a(x) \cup s_m(x));\theta)\},$$

Datasets

Domain	Dataset	#Pos	#Neg	#Neu
LAPTOP	Train	980	858	454
	Test	340	128	171
REST	Train	2159	800	632
	Test	730	195	196
TWITTER	Train	1567	1563	3127
	Test	174	174	346

Table 3: Datasets in our experiments. **#Pos**, **#Neg** and **#Neu** denotes the number of instances with Positive, Negative and Neutral sentiment, respectively.

Training Details

- Used pre-trained ***GloVe*** vectors to initialize the word embeddings with vector dimension 300
- **OOV** words: randomly sampled embeddings from uniform distribution [-0.25, 0.25]
- Initialized the other model parameters uniformly between [-0.01, 0.01]
- Overfitting: ***Dropout*** strategy
- Optimizer: ***Adam*** with learning rate 0.001
- Empirically set $\boldsymbol{K}$ to 5, $\boldsymbol{\gamma}$ as 0.1 on LAPTOP data set, 0.5 on REST data set and 0.1 on TWITTER data set, respectively.

Experiments

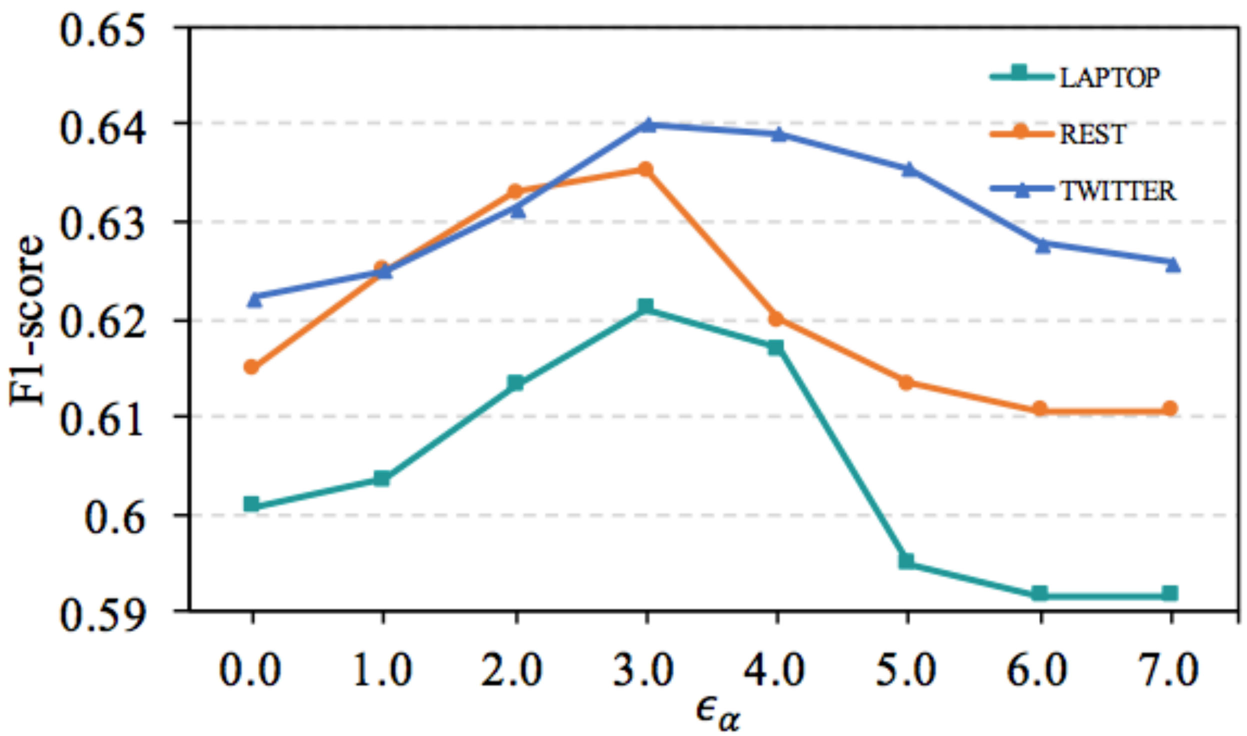


Figure 3: Effects of ϵ_α on the validation sets using MN(+AS).

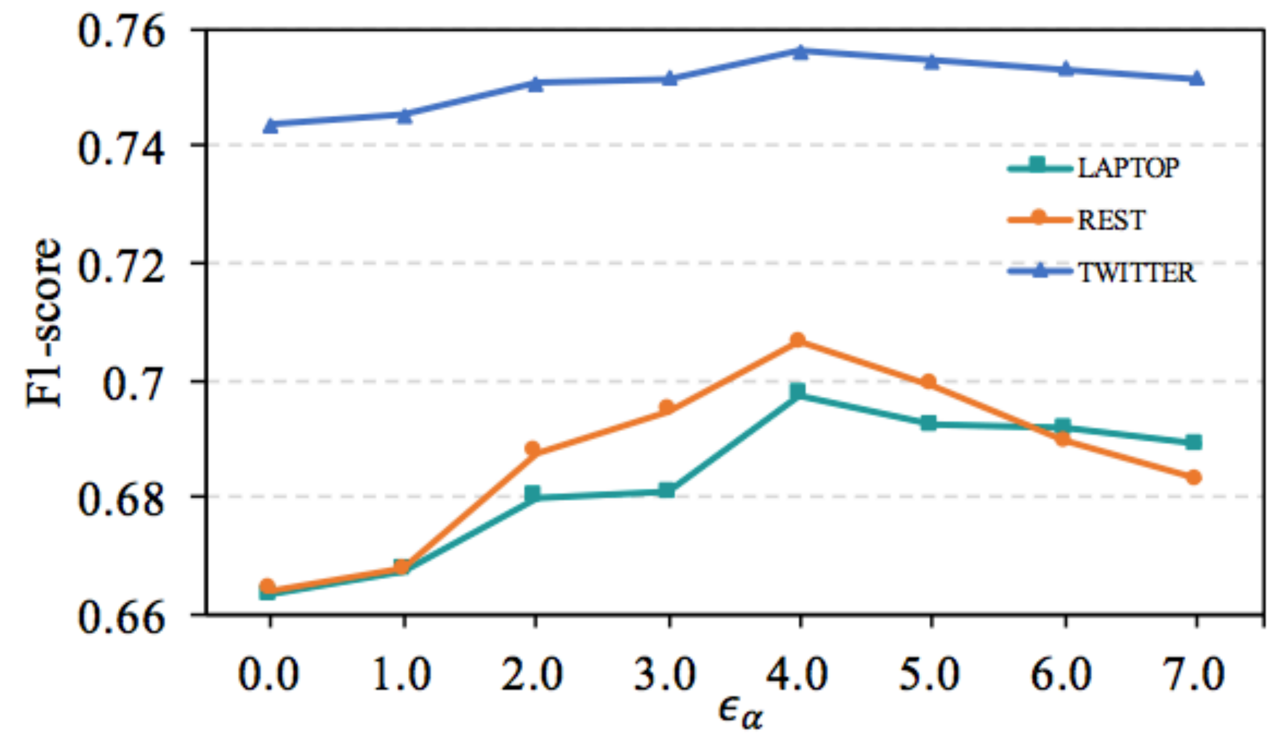


Figure 4: Effects of ϵ_α on the validation sets using TNet-ATT(+AS).

Overall Results

Model	LAPTOP		REST		TWITTER	
	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy
MN (Wang et al., 2018)	62.89	68.90	64.34	75.30	—	—
MN	63.28	68.97	65.88	77.32	66.17	67.71
MN(+KT)	63.31	68.95	65.86	77.33	66.18	67.78
MN(+AS$_m$)	64.37	69.69	68.40	78.13	67.20	68.90
MN(+AS$_a$)	64.61	69.95	68.59	78.23	67.47	69.17
MN(+AS)	**65.24****	**70.53****	**69.15****	**78.75***	**67.88****	**69.64****
TNet (Li et al., 2018)	71.75	76.54	71.27	80.69	73.60	74.97
TNet	71.82	76.12	71.70	80.35	76.82	77.60
TNet(+KT)	71.74	76.44	71.36	80.59	76.78	77.54
TNet-ATT	71.21	76.06	71.15	80.32	76.53	77.46
TNet-ATT(+KT)	71.44	76.06	71.01	80.50	76.58	77.46
TNet-ATT(+AS$_m$)	72.39	76.89	72.04	80.96	77.42	78.08
TNet-ATT(+AS$_a$)	73.30	77.34	72.67	81.33	77.63	78.47
TNet-ATT(+AS)	**73.84****	**77.62****	**72.90****	**81.53***	**77.72****	**78.61***

Table 4: Experimental results on various datasets. We directly cited the best experimental results of MN and TNet reported in (Wang et al., 2018; Li et al., 2018). $**$ and $*$ means significant at p <0.01 and p <0.05 over the baselines (MN, TNet) on each test set, respectively. Here we conducted 1,000 bootstrap tests (Koehn, 2004) to measure the significance in metric score differences.

Case Study

Model	Sentence	Ans./Pred.
TNet-ATT	The **[folding chair]** i was seated at was uncomfortable .	**Neg / Neu**
TNet-ATT(+AS)	The **[folding chair]** i was seated at was uncomfortable .	**Neg / Neg**
TNet-ATT	The **[food]** did take a few extra minutes ... the cute waiters ...	**Neu / Pos**
TNet-ATT(+AS)	The **[food]** did take a few extra minutes ... the cute waiters ...	**Neu / Neu**

Table 5: Two test cases predicted by TNet-ATT and TNet-ATT(+AS).