ALDONA: A Hybrid Solution for Sentence-Level Aspect-Based Sentiment Analysis Using a Lexicalized Domain Ontology and a Neural Attention Model

Donatas Meskele
donatas.meskele@gmail.com

Flavius Frasincar
frasincar@ese.eur.nl

Erasmus University Rotterdam
the Netherlands

Contents

Motivation

- Explosion of opinionated text on the Web (e.g., reviews)
- Difficult to manually determine opinions with respect to an entity of interest
- Sentiment Analysis (SA): automatic computation of the sentiment (e.g., positive, neutral, or negative) expressed in a piece of text (related to the entity of interest)
- Aspect-Based Sentiment Analysis (ABSA): automatic computation of the sentiment expressed for an aspect in a piece of text (related to the entity of interest)
  - Review-level
  - Sentence-level [our focus here]

Motivation

ABSA Tasks

- Target Extraction: extracting the target word or set of words present in text
- Aspect Detection: detecting the aspects from text (aspects can be broader than targets)
  - Explicit Aspect Detection: aspects have associated targets in text
  - Implicit Aspect Detection: aspects do not have associated targets in text (they are implied from text)
- Sentiment Classification: computing the sentiment associated to an (implicit or explicit) aspect [our focus here]

Related Work

ABSA Solutions

- Knowledge-Based Reasoning: using lexicalized sentiment domain ontologies one can infer the sentiment attached to an aspect in context, e.g., Ont [Schouten and Frasincar, 2018]
- Machine Learning: using machine learning approaches based on feature vectors, e.g., Bag-of-Words (BoW) [Schouten and Frasincar, 2018]
  - Deep Learning: learn feature representations and consider word order, e.g., CABASC [Liu et al., 2018]
- Hybrid: mix knowledge-based reasoning with machine learning, e.g., BoW+Ont [Schouten and Frasincar, 2018]
  - Two-Step Hybrid: first knowledge-based reasoning and then machine learning as backup, e.g., Ont+BoW [Schouten and Frasincar, 2018]

Main Ideas

- We propose a two-step hybrid approach that performs first Knowledge-Based Reasoning and then Deep Learning as backup dubbed *A Lexicalized Domain Ontology and Neural Attention (ALDONA)*
- We reuse Knowledge-Based Reasoning from Ont+Bow, but extend CABASC:
  - Employ bidirectional context attention mechanism to consider not only the forward order but also the backward order of words
  - Extend the classification module to consider the relationships between the aspect-specific sentence representation, the sentence representation, and the aspect representation
  - Relax the assumption that the number of neurons in the sentence-level content attention mechanism and the number of neurons in the classification layer have to be the same as the dimension of the word embeddings (i.e., 300)

Methodology

The methodology is based on two main steps:

1. Knowledge-Based Reasoning
2. Neural Attention Model

More precisely:

1.1 Compute the sentiment of each word in a sentence that is related to the current aspect based on a lexicalized domain sentiment ontology

1.2 If only positive sentiment found then classify the aspect as positive

1.3 If only negative sentiment found then classify the aspect as negative

2 If both positive and negative sentiment found or no sentiment found then apply the Neural Attention Model to classify the aspect as positive, negative, or neutral

Ontology

The onology has three main classes:

- *SentimentMention*: specifies the mentions of sentiment
- *SentimentValue*: specifies the polarity which can be *Positive* or *Negative*
- *AspectMention*: specifies the mentions of aspects

and two main (annotation) properties:

- *lex*: relates a mention to a lexical representation
- *aspect*: relates a sentiment to an aspect

In addition:

- The *Neutral* sentiment is not specified due to its ambiguous semantics
- We consider negation in two cases:
  - Using the dependency relation the current word is related to a *Negator*
  - One of the preceding three words with respect to the current word is a *Negator*

Sentiment Mention Types

There are three sentiment mention types:

- *Type 1*: generic sentiment mention, which has the same sentiment value for all aspects
  - e.g., "awesome" is always *Positive* (unless sarcasm present, which we do not consider here)
- *Type 2*: aspect-dependent sentiment, which has the same sentiment value for some aspects (extra check for matching the current aspect needed)
  - e.g., "delicious" is *Positive* for *SustenanceMention* (food and drinks) but does not apply to *ServiceMention*
- *Type 3*: context-dependent sentiment, which has different sentiment values for different aspects (extra check for matching the current aspect needed)
  - New axioms built based on a sentiment word linked by a dependency relation to the current aspect
  - e.g., "cold" + "beer" is *Positive* but "cold" + "pizza" is *Negative*

Neural Attention Model

The Neural Attention Model dubbed Deep Bidirectional Gated Recurrent Unit (DBGRU) has four main parts:

1. *Word Embeddings*: represents the sentence and aspect as a sequence of word embeddings
2. *Bidirectional Context Attention Mechanism*: computes word attention weights taking a local view and considering word order
3. *Sentence-Level Content Attention Mechanism*: computes word attention weights taking a global view and disregarding word order
4. *Classification Module*: classifies an aspect for sentiment (positive, neutral, or negative)

Word Embeddings

- $S = \{s_1, s_2, ..., s_N\}$ be an input sentence of length N containing words s_n
- $S_a = \{s_{i+1}, ..., s_{i+L}\}$ be an aspect a of length L in that sentence
- The embedding of a word s_n is constructed as follows:

$$e_n = \mathbb{L} o_n \in \mathbb{R}^d \tag{1}$$

where $o_n \in \mathbb{R}^{|V|}$ is a one-hot vector, $\mathbb{L}^{d \times |V|}$ is the embedding matrix, d is the length of the numeric vector associated to a word, and V is a dictionary containing all known words

- $E = [e_1, e_2, ..., e_N] \in \mathbb{R}^{d \times N}$ represents the embedded sentence
- $E_A = [e_{i+1}, ..., e_{i+L}] \in \mathbb{R}^{d \times L}$ represents the embedded aspect

Word Embeddings

We split the sentence into two parts and get:

$$\begin{aligned} S_{LS} &= \{s_1, ..., s_i, s_{i+1}, ..., s_{i+L}\} \\ S_{RS} &= \{s_{i+1}, ..., s_{i+L}, s_{i+L+1}, ..., s_N\} \end{aligned} \tag{2}$$

and their embedded versions:

$$\begin{aligned} E_{LS} &= [e_1, ..., e_i, e_{i+1}, ..., e_{i+L}] \\ E_{RS} &= [e_{i+1}, ..., e_{i+L}, e_{i+L+1}, ..., e_N] \end{aligned} \tag{3}$$

Bidirectional Context Attention Mechanism

We make use of a Gated Recurrent Unit (GRU):

$$
\begin{aligned}
r_n &= \sigma(W_r e_n + U_r h_{n-1} + b_r) \\
u_n &= \sigma(W_u e_n + U_u h_{n-1} + b_u) \\
\tilde{h}_n &= \tanh(W_h e_n + U_h(r_n \odot h_{n-1}) + b_{\tilde{h}}) \\
h_n &= u_n \odot h_{n-1} + (1 - u_n) \odot \tilde{h}_n
\end{aligned}
\tag{4}
$$

where

- $\odot$ represents the element-wise multiplication
- σ and tanh are sigmoid and hyperbolic tangent functions
- $e_n \in \mathbb{R}^d$ is a word embedding vector, $r_n \in \mathbb{R}^d$ and $u_n \in \mathbb{R}^d$ are the reset and update gates
- $\tilde{h}_n \in \mathbb{R}^d$ and $h_n \in \mathbb{R}^d$ are the new memory and the new hidden state
- $W_r \in \mathbb{R}^{d \times d}$, $U_r \in \mathbb{R}^{d \times d}$, $W_u \in \mathbb{R}^{d \times d}$, $U_u \in \mathbb{R}^{d \times d}$, $W_h \in \mathbb{R}^{d \times d}$, $U_h \in \mathbb{R}^{d \times d}$ are weight matrices and $b_r \in \mathbb{R}^d$, $b_u \in \mathbb{R}^d$, $b_{\tilde{h}} \in \mathbb{R}^d$ are bias vectors

Bidirectional Context Attention Mechanism

More precisely we use a bidirectional GRU (BGRU):

$$\begin{aligned}
\overrightarrow{h_n} &= f(\overrightarrow{\Theta} |\ e_n, \overrightarrow{h_{n-1}}) \\
\overleftarrow{h_n} &= f(\overleftarrow{\Theta} |\ e_n, \overleftarrow{h_{n+1}}) \\
h_n &= tanh(W_{fw}\overrightarrow{h_n} + W_{bw}\overleftarrow{h_n} + b_{bi})
\end{aligned} \tag{5}$$

where

- $\overrightarrow{h_n}$ and $\overleftarrow{h_n}$ are hidden states obtained from the forward and backward directions
- e_n is the new input
- $\overrightarrow{\Theta}$ and $\overleftarrow{\Theta}$ are parameters to be optimised (here, two sets of weight matrices and bias vectors described in Eq. 4)
- $W_{fw} \in \mathbb{R}^{d \times d}$ and $W_{bw} \in \mathbb{R}^{d \times d}$ are weight matrices and $b_{bi} \in \mathbb{R}^{d}$ is a bias vector

Bidirectional Context Attention Mechanism

- The left and right part embeddings, E_{LS} and E_{RS} are separately fed to the bidirectional gated recurrent unit (BGRU) and produce the outputs:

$$\begin{aligned} H_{LS} &= [h_L^l, ..., h_{i+1}^l, h_i^l, ..., h_1^l] \\ H_{RS} &= [h_{i+1}^r, ..., h_L^r, h_{L+1}^r, ..., h_N^r] \end{aligned} \tag{6}$$

- By exploiting this information and employing a Feedforward Neural Network (FNN) we get bidirectional context attention weights for each word in the sentence S:

$$\begin{aligned} \beta_i^l &= \sigma(W_1 h_i^l + b_1) + b_l \\ \beta_i^r &= \sigma(W_2 h_i^r + b_2) + b_r \end{aligned} \tag{7}$$

where $W_1 \in \mathbb{R}^{1\times d}$ and $W_2 \in \mathbb{R}^{1\times d}$ are weight matrices, $b_1 \in \mathbb{R}$ and $b_2 \in \mathbb{R}$ are biases, and $b_l \in \mathbb{R}$ and $b_r \in \mathbb{R}$ are basic attention weights

Bidirectional Context Attention Mechanism

- The attention weights β (one scalar attention weight per word) are defined as follows:

$$
\begin{aligned}
\beta_{LS} &= [\beta_1^l, ..., \beta_{i+1}^l, ..., \beta_L^l] \\
\beta_{RS} &= [\beta_{i+1}^r, ..., \beta_L^r, ..., \beta_N^r] \\
\beta_A &= [\frac{\beta_{i+1}^l + \beta_{i+1}^r}{2}, ..., \frac{\beta_L^l + \beta_L^r}{2}] \\
\beta_{LC} &= [\beta_1^l, ..., \beta_i^l] \\
\beta_{RC} &= [\beta_{i+L+1}^r, ..., \beta_N^r] \\
\beta &= [\beta_{LC}, \beta_A, \beta_{RC}]
\end{aligned}
\tag{8}
$$

- Each memory slice $m_{w_n} \in \mathbb{R}^d$ of the weighted memory $M_w = [m_{w_1}, ..., m_{w_N}]$ is constructed as follows:

$$
m_{w_n} = \beta_{tiled} \odot e_n \tag{9}
$$

where $\beta_{tiled} \in \mathbb{R}^d$ is the element $\beta_n \in \mathbb{R}$ replicated d times, and $\odot$ represents the element-wise multiplication

Sentence-Level Content Attention Mechanism

- We compute an aspect representation v_a and a sentence representation v_s as follows:

$$v_a = \frac{1}{L}\sum_{l=1}^{L} e_l \qquad v_s = \frac{1}{N}\sum_{n=1}^{N} e_n \tag{10}$$

- We compute the attention score c_n for each word s_n in the sentence as follows:

$$c_n = W_3 \tanh(W_4 m_{w_n} + W_5 v_a + W_6 v_s + b_3) \tag{11}$$

where $m_{w_n} \in \mathbb{R}^d$ is the weighted memory slice of the word s_n, $v_a \in \mathbb{R}^d$ and $v_s \in \mathbb{R}^d$ are the aspect and sentence representations, $W_3 \in \mathbb{R}^{1\times m}$, $W_4 \in \mathbb{R}^{m\times d}$, $W_5 \in \mathbb{R}^{m\times d}$, $W_6 \in \mathbb{R}^{m\times d}$ are weight matrices and $b_3 \in \mathbb{R}^m$ is a bias vector

Sentence-Level Content Attention Mechanism

- The attention weight α_n for each word s_n is determined as follows:

$$\alpha_n = exp(c_n) / \sum_{j=1}^{N} exp(c_j) \tag{12}$$

- A weighted embedding sentence vector $v_{we} \in \mathbb{R}^d$ is calculated by:

$$v_{we} = M_w \alpha \tag{13}$$

where $\alpha = [\alpha_1, ..., \alpha_N]^T \in \mathbb{R}^N$, $M_w \in \mathbb{R}^{d \times N}$

Classification Module

- We define the output vector $v_o \in \mathbb{R}^k$ that models the relationship between the sentence representation v_s, the aspect representation v_a, and the weighted embedding sentence vector v_{we} as follows:

$$
\begin{aligned}
v_{sw} &= \tanh(W_7 v_s + W_8 v_{we} + b_4) \\
v_{aw} &= \tanh(W_9 v_a + W_{10} v_{we} + b_5) \\
v_o &= \tanh(W_{11} v_{sw} + W_{12} v_{aw} + b_6)
\end{aligned} \tag{14}
$$

where $W_7 \in \mathbb{R}^{d \times d}$, $W_8 \in \mathbb{R}^{d \times d}$, $W_9 \in \mathbb{R}^{d \times d}$, $W_{10} \in \mathbb{R}^{d \times d}$, $W_{11} \in \mathbb{R}^{k \times d}$, and $W_{12} \in \mathbb{R}^{k \times d}$ are weight matrices, $b_4 \in \mathbb{R}^d$, $b_5 \in \mathbb{R}^d$, and $b_6 \in \mathbb{R}^k$ are bias vectors, $v_a \in \mathbb{R}^d$ and $v_s \in \mathbb{R}^d$ are the aspect and sentence representations, and $v_{we} \in \mathbb{R}^d$ is the weighted embedding sentence vector

Classification Module

- A linear layer is used to convert the output vector $v_o \in \mathbb{R}^k$ into a vector $v_L \in \mathbb{R}^{|C|}$:

$$v_L = W_{13} v_o + b_7 \tag{15}$$

where $|C|$ is the number of possible aspect polarity categories ($|C|$=3 here), $W_{13} \in \mathbb{R}^{|C| \times k}$ is a weight matrix and $b_7 \in \mathbb{R}^{|C|}$ is a bias vector

- The linear layer output v_L is fed into a *softmax* function to generate aspect's polarity probabilities $p \in \mathbb{R}^{|C|}$:

$$p = softmax(v_L) \tag{16}$$

Classification Module

- We minimise the cross-entropy loss function given below:

$$loss = -\sum_{C}\sum_{S} y_{c,s} \ln(p_{c,s}) \tag{17}$$

  where C is the set of polarity categories, S are the training examples, $p_{c,s} \in [0, 1]$ is the estimated probability that a given aspect in a sentence s belongs to a category c, and $y_{c,s} \in \mathbb{B}$ is the true probability that the aspect in the sentence s is in the category c

- We apply the dropout technique to reduce model complexity and prevent overfitting

Data

- SemEval 2016 Task 5
  - Subtask 1
    - Restaurant Domain English Training Data
    - Restaurant Domain English Gold Annotations Data
- Web restaurant reviews
- Example

```
<sentence id="en_BlueRibbonSushi_478218345:2">
 <text>It has great sushi and excellent service.</text>
 <Opinions>
   <Opinion target="sushi" category="FOOD#QUALITY"
            polarity="positive" from="13" to="18" />
   <Opinion target="service" category="SERVICE#GENERAL"
            polarity="positive" from="35" to="42" />
 </Opinions>
</sentence>
```

Data

- Polarity distributions in train and test data sets:

	Positive		Neutral		Negative		Total	
	Freq.	%	Freq.	%	Freq.	%	Freq.	%
Train data	1314	70.19	71	3.79	487	26.02	1872	100
Test data	478	74.34	32	4.98	133	20.68	643	100

- Data is skewed towards positive sentiment
- The most dominant aspect is "FOOD#QUALITY" (around 40% of occurences)

Data Preprocessing:

- Pure train and validation data sets are created by stratified random sampling, and by splitting the train data set into 75/25 proportions
- We applied tokenization and lemmatization for inferring the word types used in Knowledge-Based Reasoning

Data

Data Cleaning:

- All words are converted to lowercase, “"”, “'” and “&” are replaced with a double quote symbol (”), an apostrophe (’), and the word *and*
- All punctuation signs, numbers, and tabulation are removed
- Sentences containing implicit aspects (*target*=*“NULL”*) are not considered (as we need the targets)
- Sentences without *Opinions* are also excluded (no use)

Word Emebeddings:

- GloVe word embedding vectors: 1.9 million vocabulary size with 300-dimensional vectors
- We eliminate words without embedding vectors from train and test data sets

Evaluation

We evaluate the following models:

- *Ont*: Knowledge-Based Reasoning with backup majority polarity (from [Schouten and Frasincar, 2018])
- *BaseA*: content attention (from [Liu et al., 2018])
- *BaseB*: sentence-level content attention (from [Liu et al., 2018])
- *BaseC*: sentence-level position attention (from [Liu et al., 2018])
- *CABASC*: sentence-level context attention with GRU (from [Liu et al., 2018])
- *CTX-LSTM*: sentence-level context attention with LSTM
- *CTX-BLSTM*: sentence-level context attention with BLSTM
- *CTX-BGRU*: sentence-level context attention with BGRU
- *DBGRU*: neural attention model of ALDONA
- *ALDONA*: our proposed model

Evaluation

- Train and test classification accuracy:

	Train accuracy	**Test accuracy**
Ont	74.95	78.38
BaseA	77.40	83.05
BaseB	83.76	84.60
BaseC	84.29	85.23
CABASC	79.65	84.45
CTX-LSTM	82.43	84.60
CTX-BLSTM	84.99	84.60
CTX-BGRU	86.65	85.38
DBGRU	89.58	86.00
ALDONA	90.17	86.31

Evaluation

Configuration of TensorFlow (ALDONA software is available at `https://github.com/donmesh/ALDONA`):

- The basic hyperparameters, namely, $b_r = b_l = 0.5$, $d = 300$, $learning\_rate = 0.001$, and normally ($N(0, 0.0025)$) initialised weight matrices used for all models were inherited from Liu et al. [2018]
- ALDONA $dropout\_probability = 0.3$ (using the validation dataset)
- ALDONA specific hyperparameters are set based on the grid search ($0.5d$, d, and $2d$, where $d = 300$): $m = 300$ (from Eq. 11) and $k = 150$ (from Eq. 14) (using the validation dataset)
  - CABASC uses only one variable $d = 300$
- For efficiency pruposes we use the Minibatch Gradient Descent algorithm with $batch\_size = 128$ set based on the grid search (64, 128, and 256) (using the validation dataset)
  - CABASC uses the Stochastic Gradient Descent algorithm

Concluding Remarks

Conclusion

- We have proposed ALDONA, a two-step hybrid method for sentence-level aspect-based sentiment analysis:
  1. Knowledge-Based Reasoning
  2. Neural Attention Model
- Obtained competitive results with respect to the state-of-the-art CABASC method

Future Work

- Develop a learning algorithm for the lexicalized domain sentiment ontology (reuse work on ontology learning)
- Propose a solution to deal with implicit aspects (finding target proxies based on word similarity)

References

Qiao Liu, Haibin Zhang, Yifu Zeng, Ziqi Huang, and Zufeng Wu. Content Attention Model for Aspect Based Sentiment Analysis. In *Proceedings of the 2018 World Wide Web Conference (WWW 2018)*, pages 1023–1032. IW3C2, 2018.

Kim Schouten and Flavius Frasincar. Ontology-Driven Sentiment Analysis of Product and Service Aspects. In *Proceedings of the 15th Extended Semantic Web Conference (ESWC 2018)*, volume 10360 of *LNCS*, pages 608–623. Springer International Publishing, 2018.