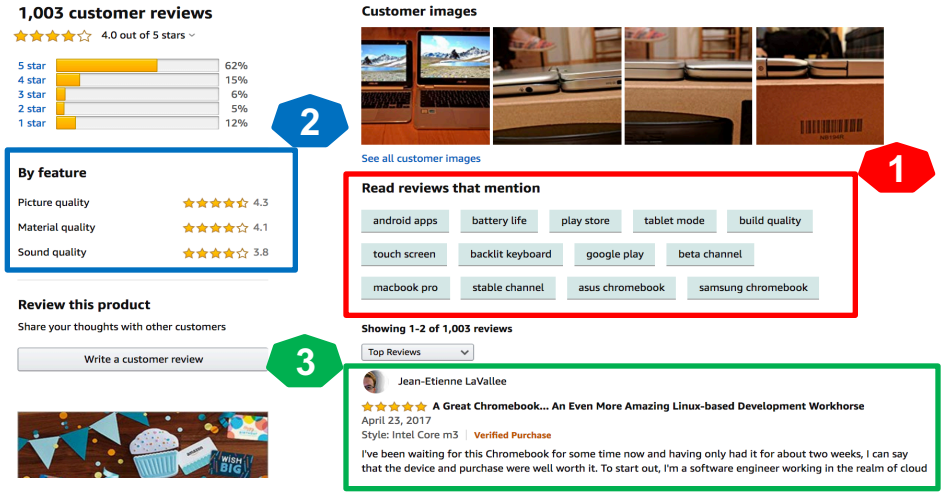
Aspect Term Extraction in Product Reviews

Subrata Ghosh

Outline

- **Project description**
- **Data: Statistics**
- **Logics and results**
  - **Logic – 1:** Unsupervised approach(Rules based)
  - **Logic – 2:** Supervised approach(Bag of aspect)
  - **Logic – 3:** Logic – 1 + Logic - 2
- **Future work**

Use case



General use case

Case 1: Find most important aspects.
Aspect based review filtering.

Case 2: Aspect based rating.

Case 3: Combine customer review based on aspect terms.

PMI use case

Case 1: New marketing strategy.
Case 2: Product improvement.

Problem Definition

Aspect Term(AT):

- Opinionated expressions.

Definition

- It is the task of automatically extract aspect term from product review text.
- For each word of a review, the model should predict if the word is an aspect term or not of the reviewed product.
- It is a **multiclass** classification problem. Multiclass = {**O** : not an aspect, **B** : first word of an aspect (Beginning), **I** : second, third, ... word of an aspect (Inside)}
- It is a subtask of aspect-based sentiment analysis.

What need to do?

- **Given:** Product reviews and aspect terms
- **Need to do:** Build an AT extractor model which should predict the word is a class {O, B, I} of the reviewed product.

Problem: Example

Consider the product review and aspect terms:

Review text: The **battery life** is really good and its **size** is reasonable

Aspect terms: Battery life, size.

BIO format:

The	battery	life	is	really	good	and	its	size	is	reasonable
O	B	I	O	O	O	O	O	B	O	O

ATE is to assign to each word one of the three possible classes:

- O = not an aspect (Outside)
- B = first word of an aspect
- I = second, third, … word of an aspect (Inside)

Applications

1. Find popular, positive, negative aspects of a particular product.
2. Improve customer experiences.
3. Contribute marketing strategy and product improvement.
4. Can complement information retrieval, text summarization etc.

Challenges

Consider the Laptop reviews:

Examples:

1. I expected so as it's an **Apple product**.
2. I like **Firefox**.
3. The **battery life** is really good and its size is reasonable.

Challenge 1: AT is opinionated expression. See example 1.

Challenge 2: Domain product aspect terms. See example 2.

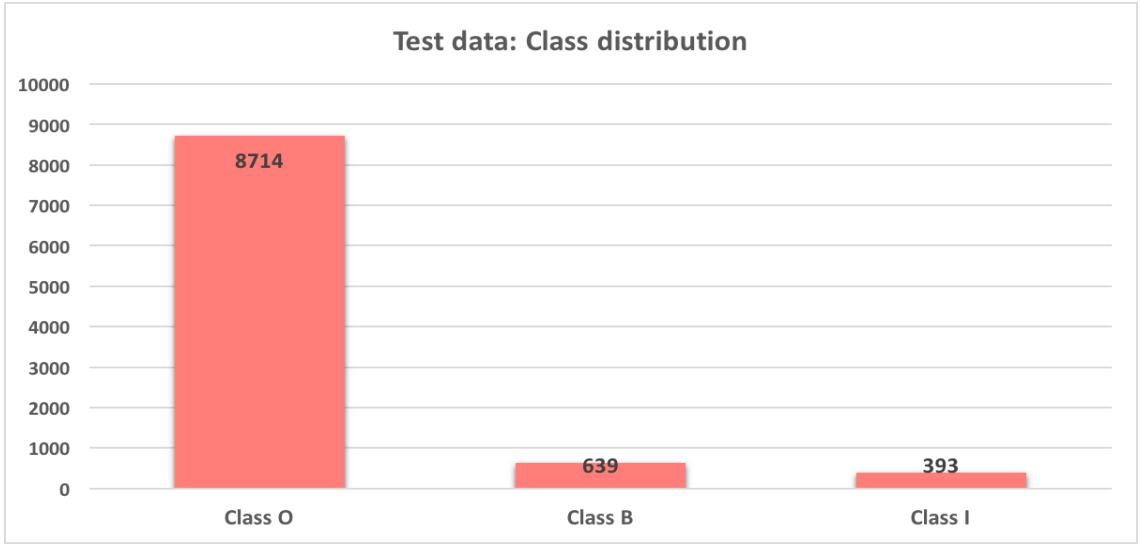
Challenge 3: Aspect term can be composed of more than one token. See example 3

Outline

- Project description
- **Data: Statistics**
- Logics and results
  - **Logic – 1:** Unsupervised approach(Rules based)
  - **Logic – 2:** Supervised approach(Bag of aspect)
  - **Logic – 3:** Logic – 1 + Logic - 2
- Future work

Data: Statistics

Data type	Number of review	#aspect (single word)	#aspect (multi word)
Training	3045	365	590
Test	800	167	226



Outline

- Project description
- Data: Statistics
- **Logics and results**
  - **Logic – 1:** Unsupervised approach(Rules based)
  - **Logic – 2:** Supervised approach(Bag of aspect)
  - **Logic – 3:** Logic – 1 + Logic - 2
- Future work

Logic 1: Unsupervised approach

- I try to solve this problem completely unsupervised way. I am not use training data for this model.
- Extract aspect terms using syntactic dependencies of words in a sentence.

Workflow with example

Review text: The battery life is really good and its size is reasonable

↓

Aspect - opinion pair: [(battery, good), (size , reasonable), **(battery, life)**]

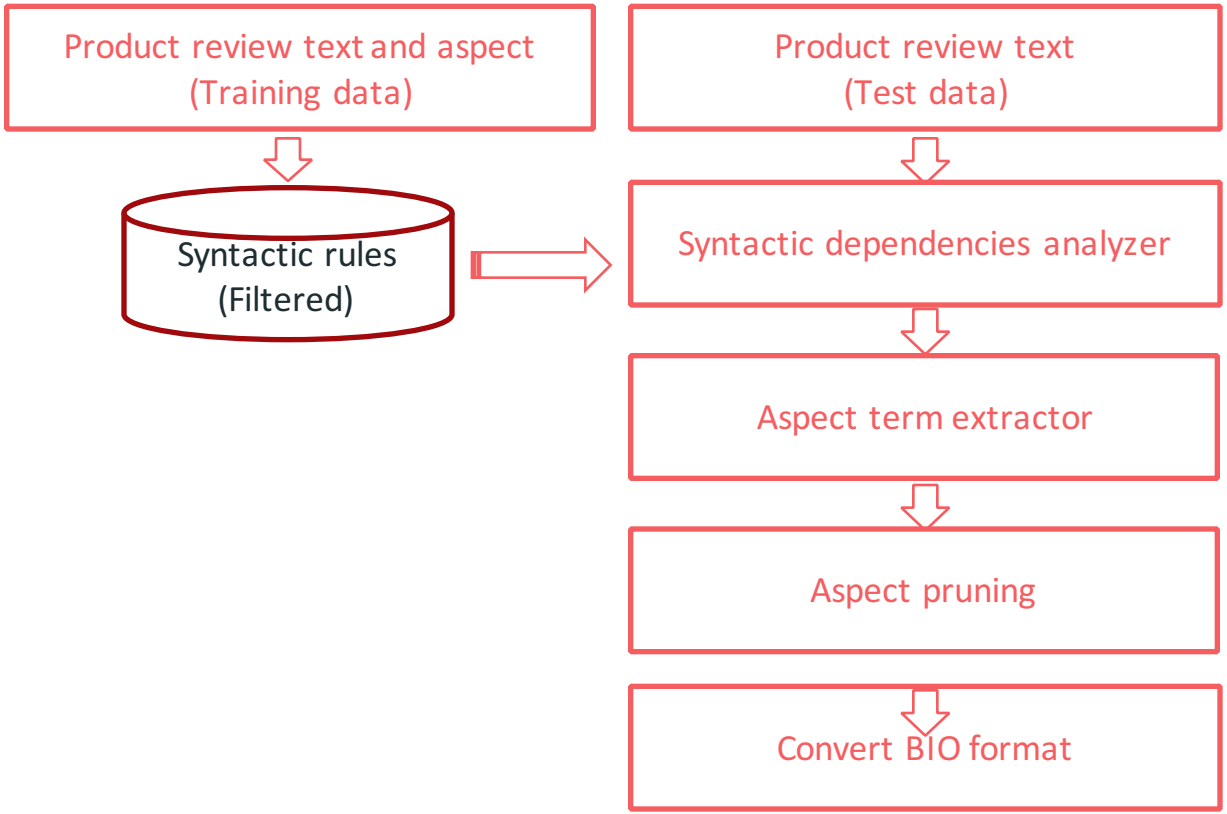
↓

Aspect terms : [battery life, size]

↓

Convert to BIO format: [(The, O), (battery, B), (life, I), (is, O), (really, O), (good, O), (and, O), (its, O), (size, B), (is, O), (reasonable, O)]

Logical workflow



Algorithm: Syntactic rules

1. **Nominal subject:** The aspect is the subject of the opinion.
2. **Noun compound modifier:** Noun that serves to modify the head noun.
3. **Adjectival modifier:** The opinion is an adjectival modifier of the aspect.
4. **Direct object:** The target is the direct object of the opinion.

From training data, I find what syntactic rules and pattern can be useful.

Stanford Parser: In build syntactic rules tool.

Stanford Parser notation:

(('good', 'JJ'), 'nsubj', ('life', 'NN'))
(('reasonable', 'JJ'), 'nsubj', ('size', 'NN'))
(('life', 'NN'), 'compound', ('battery', 'NN'))

From above notations we an create aspect-opinion pair.

This type of several syntactic rules we use to create aspect-opinion pair.

Algorithm: Aspect terms extractor

Consider the product review and aspect terms:

Review text: The battery life is really good and its size is reasonable

Stanford Parser notation	Extracted aspect
(('good', 'JJ'), **'nsubj'**, ('life', 'NN'))	life
(('reasonable', 'JJ'), **'nsubj'**, ('size', 'NN'))	size
(('life', 'NN'), **'compound'**, ('battery', 'NN'))	battery life

Final aspect terms: {battery life, size}

In the above text **“good”** and **“reasonable”** is opinion word and **“battery life”** and **“size”** is aspects.

Algorithm: Aspect pruning

Some cases, redundant aspect could be extracted, let's consider below aspect term list:

Extracted aspect terms: [**battery life**, **life**, size]

Now, you can see “life” aspect already included in “battery life”.

After removing redundant aspect: [battery life, life, size]

Evaluation

Logic	Class	Precision	Recall	F1-score
Logic-1	O	0.94	0.89	0.91
Logic-1	B	0.23	0.43	0.30
Logic-1	I	0.34	0.19	0.25

This is a class imbalance problem, as you can see in the data set almost 90% of word belongs to class O.

Class I: recall is very low, that mean this model only 19% correctly find out. Also, precision not good.

Class B: recall is not bad but precision is too low, that means wrong classification is high

Class O: Just because of class imbalance results looks good but actually it is not.

Outline

- Project description
- Data: Statistics
- **Logics and results**
  - **Logic – 1:** Unsupervised approach(Rules based)
  - **Logic – 2:** Supervised approach(Bag of aspect)
  - **Logic – 3:** Logic – 1 + Logic - 2
- Future work

Logic 1: Pro and Cons

Pro:

1. Totally unsupervised, no need any training data.

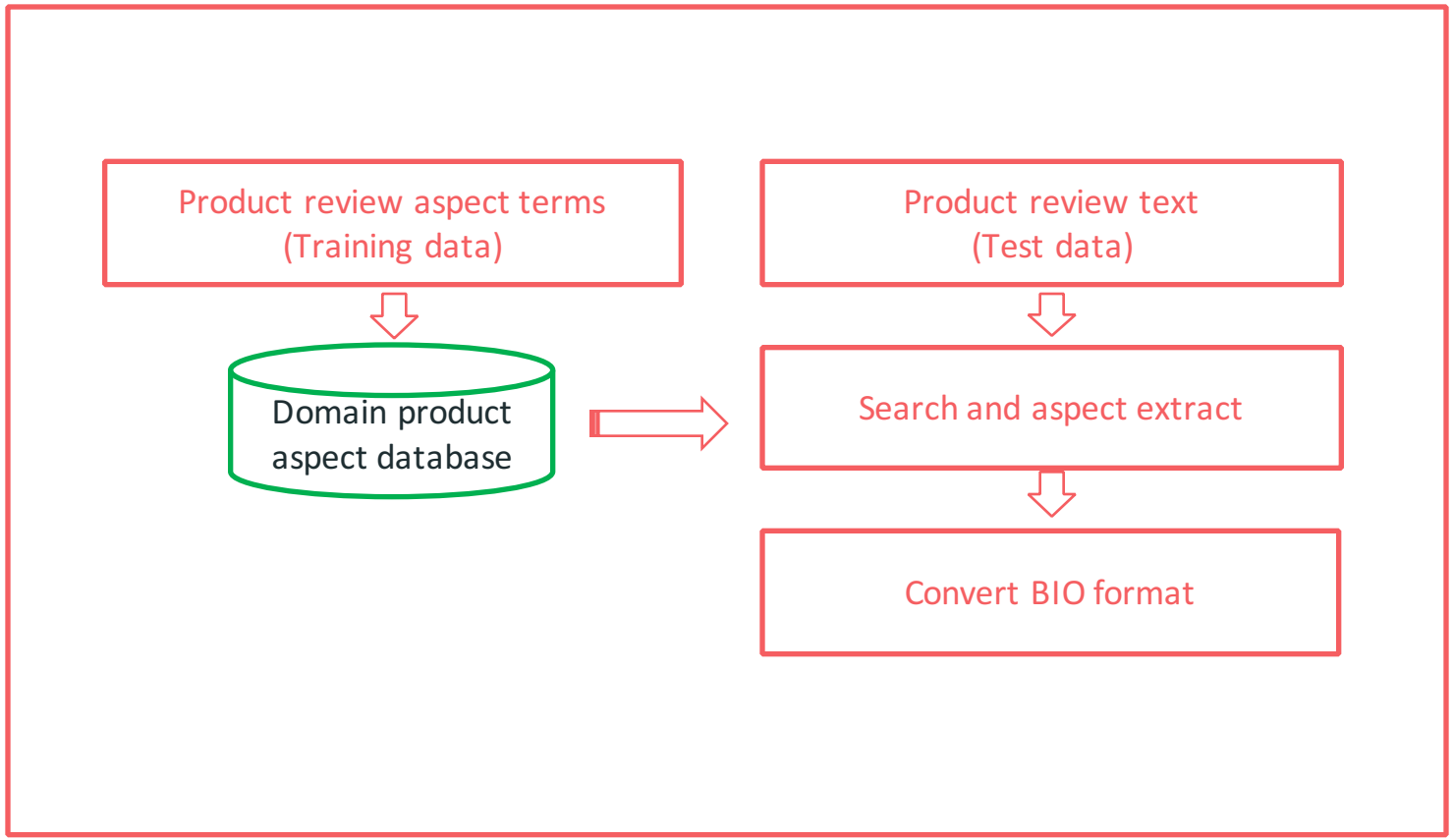
Cons:

1. Not consider domain product aspect(challenge 2).
2. Class I: recall is too low.

Logic 2: Supervised approach

- This approach is completely supervised. I use training data for this model.
- It is string search based approach.

Logical workflow



Algorithms

Domain product aspect database(DPAD) creation: In training data already given aspect terms. I use all those aspects to find aspect in test data.

Consider below review and DPAD:

- **DPAD:** [“battery life” ,“size”,……..]
- **Review text:** The battery life is really good and its size is reasonable

After search aspect terms = [battery life, size]

Aspect search in test data:

- Just take each aspect term from DPAD and search it in the test review, if that aspect present in the review take it as a aspect term of that review.

Evaluation

Logic	Class	Precision	Recall	F1-score
Logic-1	O	0.94	0.89	0.91
Logic-1	B	0.23	0.43	0.30
Logic-1	I	0.34	0.19	0.25
Logic-2	O	0.96	0.95	0.96
Logic-2	B	0.45	0.74	0.56
Logic-2	I	0.95	0.14	0.24

Class I: recall is very low, that mean this model only 14% correctly find out. But, precision is too good.

Class B: recall is good and precision aslo better then logic-2

Outline

- Project description
- Data: Statistics
- **Logics and results**
  - **Logic – 1:** Unsupervised approach(Rules based)
  - **Logic – 2:** Supervised approach(Bag of aspect)
  - **Logic – 3:** Logic – 1 + Logic - 2
- Future work

Logic 2: Pro and Cons

Pro:

1. Results is good compere to other logic.
2. Based on domain product aspect.

Cons:

1. Completely wrong approach, because it not fulfill aspect term definition itself.
2. Failed to solve challenge-1.
3. Class I: recall is very very low.

Logic 3: Combination of Logic-1 and Logic-2

Observation:

- **Logic-1:** It is not use domain product aspect and results is not good.
- **Logic-2:** Not fulfill AT definition so it is wrong approach although it gives good results. Class –I, recall is very low.

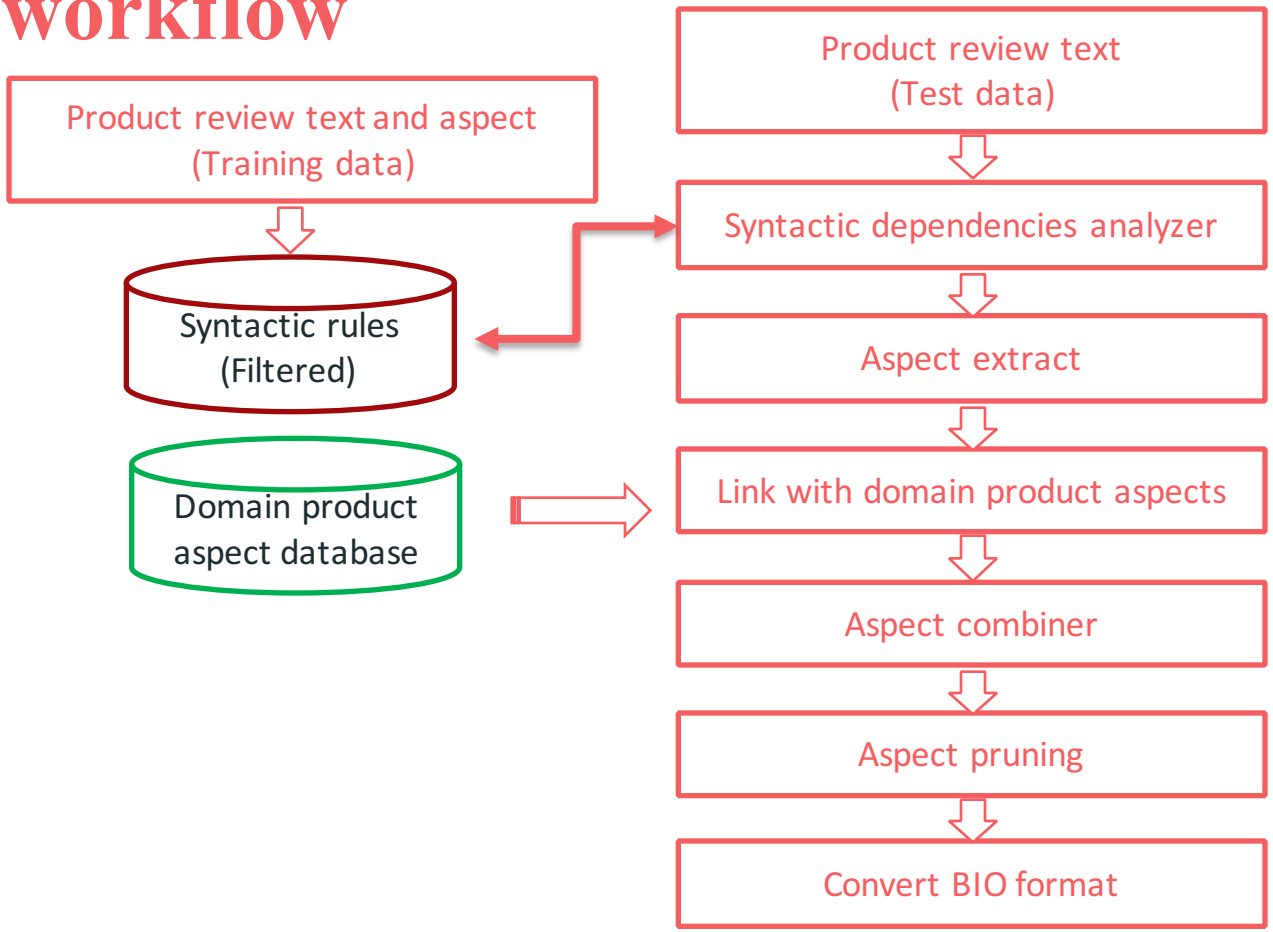
Goal:

- Remove all the cons of those two logics and get good results.

Idea:

- Combine both the logic.

Logical workflow



Algorithm: Syntactic rules

1. **Nominal subject:** The aspect is the subject of the opinion.
2. **Noun compound modifier:** Noun that serves to modify the head noun.
3. **Adjectival modifier:** The opinion is an adjectival modifier of the aspect.
4. **Direct object:** The target is the direct object of the opinion.

From training data, I find what syntactic rules and pattern can be useful.

Stanford Parser: In build syntactic rules tool.

Stanford Parser notation:

(('good', 'JJ'), 'nsubj', ('life', 'NN'))
(('reasonable', 'JJ'), 'nsubj', ('size', 'NN'))
(('life', 'NN'), 'compound', ('battery', 'NN'))

From above notations we an create aspect-opinion pair.

This type of several syntactic rules we use to create aspect-opinion pair.

Algorithm: Aspect terms extractor

Consider the product review and aspect terms:

Review text: The battery life is really good and its size is reasonable

Stanford Parser notation	Extracted aspect
(('good', 'JJ'), **'nsubj'**, ('life', 'NN'))	life
(('reasonable', 'JJ'), **'nsubj'**, ('size', 'NN'))	size
(('life', 'NN'), **'compound'**, ('battery', 'NN'))	battery life

Final aspect terms: {battery life, size}

In the above text **“good”** and **“reasonable”** is opinion word and **“battery life”** and **“size”** is aspects.

Algorithm: Link with domain aspect

- All aspects got from aspect extractor just filter it.
- Take only those aspects, also present in domain product aspects database.

Algorithm: Aspect combiner

Aspect term can be composed of more than one token so how I deal with?

Syntactic dependency rule :

- (('life', 'NN'), **'compound',** ('battery', 'NN')) ⇒ **battery life**

Sentence POS tag sequence:

1. **Noun followed by Noun:**
   - [('The', 'DT'), **('battery', 'NN'), ('life', 'NN'),** ('is', 'VBZ'),('really', 'RB'),('good', 'JJ')]. ⇒ **Battery life**
2. **Noun followed by Cardinal number:**
   - [('I', 'PRP'), ('like', 'VBP'), **('windows', 'NNS'), ('8', 'CD')**]. ⇒ **Windows 8**
3. **Noun before Noun**

Algorithm: Aspect pruning

Some cases, redundant aspect could be extract, lets consider below aspect term list:

Extracted aspect terms: [**battery life**, **life**, size]

Now, you can see “life” aspect already included in “battery life”.

After removing redundant aspect: [battery life, life, size]

Last: convert to BIO format and evaluate

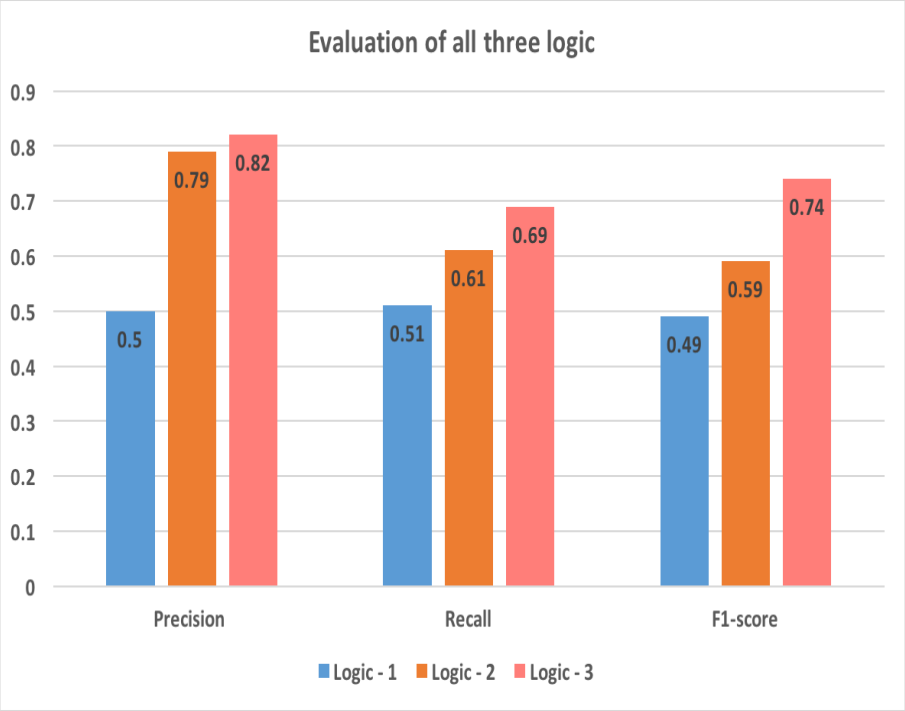
Evaluation

Logic	Class	Precision	Recall	F1-score
Logic-1	O	0.94	0.89	0.91
Logic-1	B	0.23	0.43	0.30
Logic-1	I	0.34	0.19	0.25
Logic-2	O	0.96	0.95	0.96
Logic-2	B	0.45	0.74	0.56
Logic-2	I	0.95	0.14	0.24
Logic-3	O	0.96	0.99	0.97
Logic-3	B	0.72	0.64	0.68
Logic-3	I	0.79	0.43	0.56

Class I: Now recall is reasonable and precision is good.

Class B: recall is good and precision also better then logic-2

Evaluation: All three logic



- In this multi-class setting, I take Micro-average Method.
- Logic – 3 performs best in terms of every evaluation metrics.

Outline

- **Project description**
- **Data: Statistics**
- **Logics and results**
  - **Logic – 1:** Unsupervised approach(Rules based)
  - **Logic – 2:** Supervised approach(Bag of aspect)
  - **Logic – 3:** Logic – 1 + Logic - 2
- **Future work**

Features work

1. **Syntactical relations:**

   - Can be improved further on by looking up more combined patterns and I have a gut feeling there are more combined patterns out there.

2. **Aspect term with multiple token:**

   - Yes, this part also we can improve by adding more POS tag pattern.

Thank You