Yiqun Jiang z5129432
Sichen Li z5119193

COMP9318 Project Report

Introduction

In this project, we implemented a binary classifier, which belongs to the SVM family, to train data. The training data is a sample of 540 paragraphs, 180 for class-1, and 360 for class-2. After training, we used this classifier to modify test data, which is a sample of 200 paragraphs. We changed exactly 20 places for each paragraph. By modifying, most of these paragraphs would changes their classified class from class-1 to class-0.

Basic Methodology

There are several steps in this project. For each step, it may have many different approaches, but we only have limited attempts to get feedback. Therefore, to solve this problem, we have a basic methodology for getting feedback. Unless we have reasons not to do so.

- Rule 1: Use default parameter rather than changing it
- Rule 2: Take easy approach rather than complicated one

Main Steps

This project can be divided into three main steps, which are feature extraction, SVM training and word modification.

1. feature extraction

To begin with, according to Text feature extraction, we need to mainly consider three

parts in feature extraction.

1) how to count tokens

- count the frequency of words (the number of times of a word appear in a paragraph)
  - this is the default parameter, so we want to try it
- count the occurrence of words (1 for occurrence 0 otherwise)
  - this is a reasonable choice, because we know that in word modification, removing all occurrence of a word only count once, we think this may imply that we should only count the occurrence of words in feature extraction.

2) how to tokenize

- only consider 2 or more letter words
  - this is the default parameter, but this method will ignore all punctuation, including '.' in 'Mr.', so we discard this method.
- only consider 2 or more letter characters
  - we want to try this because this is very similar to default setting
- consider all characters
  - we want to try this because this is a straightforward approach after the former one failed.

3) how to normalize

- not use Tf-idf and no normalization
  - we want to try this because it looks simply
- using Tf-idf with l2 normalization
  - we want to try this because we want to reweighing high frequency words, according to Tf–idf term weighting, this is quite reasonable choice in text feature extraction

2. SVM training

The main function of SVM training was given by project specification. According to the specification, we need to consider the following parameters:

1) kernel

although 'rbf' is the default setting, only 'linear' kernel has the weights assigned to the features, which is critical in word changing step. So, we use 'linear' kernel.

2) gamma

not available in 'linear' kernel

3) C

1.0 is the default value, but other value or even a grid search is good to try

4) degree

not available in 'linear' kernel

5) coef0

not available in 'linear' kernel

3. word modification

We use the weight of words which we got from SVM training step to modify test-data.txt. Basically, we remove the most distinguished class1 words and add some most distinguished class0 words.

We can think of two different approaches:

1) static word change (remove and add a fixed number of words)

- this is a easy approach, in this method, we have another parameter called magic_number. We remove magic_number of most distinguished class1 words and then add 20-magic_number of most distinguished class0 words.
- we call this magic_number because even we have some understanding of how this number would affect our result, we still don't have a solid mathematic way to calculate what is the best magic_number
- we try magic_number=10 because we think this is the most straightforward number to begin with.

2) dynamic word change (remove and add based on the weight)

- in this approach, every time when we remove or add words, we will compare

whether removing or adding will contribute more in making our paragraphs more like class0. We implement an algrithum to do this based on the weight of words which we got from SVM training step.
- We want to try this, because this seems the best approach theoretically.

Results

1. Results in feature extraction

We find that the following combination has the best performance

- count the frequency of words
- consider all characters
- using Tf-idf with l2 normalization

2. Results in SVM training

A 'linear' kernel with a fix default C has a reasonable (even may not the best) performance

3. Results in word modification

- static word change with magic_number=10 has a normal performance
- dynamic word change has a very poor performance

Further Results and Conclusions

The result in word modification is quite interesting, because theoretically, a dynamic word change method seems works better. But in fact, it works worse than a static word change with magic_number=10.

Our understanding is that because we have so many features and relatively small number of training samples. The model may end up with an over-fitting situation. This

mean that our weight list is not correct, there would be lot of noise words in it. Those words have very high or very low weight, but in fact they should have around 0 weight.

If we add these words to test-data.txt, they won't contribute to make it more like class-0. Instead, if we remove more words, it would work better because the words we removed must be in the paragraph first.

This can explain why dynamic word change has a very poor performance. The reason is in fact it will add more words than remove. Also, it shows that we should remove more words, so we test different magic_number and get following result:

- static word change with magic_number=20
  - This is good to try if removing is better than adding
- static word change with magic_number=15
  - we just choose a number between 15 and 20, because we think the words have highest weight may not be noise.

The result shows that this two magic_number have very similar performance. Because we think in the final test, professor may use a larger test set (we are not so sure), the over-fitting problem may not be so serious. Therefore, static word change with magic_number=15 may have a better performance than static word change with magic_number=20.

COMP9318 Project Report

Introduction

In this project, we implemented a binary classifier, which belongs to the SVM family, to train data. The training data is a sample of 540 paragraphs, 180 for class-1, and 360 for class-2. After training, we used this classifier to modify test data, which is a sample of 200 paragraphs. We changed exactly 20 places for each paragraph. By modifying, most of these paragraphs would changes their classified class from class-1 to class-0.

Basic Methodology

There are several steps in this project. For each step, it may have many different approaches, but we only have limited attempts to get feedback. Therefore, to solve this problem, we have a basic methodology for getting feedback. Unless we have reasons not to do so.

- Rule 1: Use default parameter rather than changing it
- Rule 2: Take easy approach rather than complicated one

Main Steps

This project can be divided into three main steps, which are feature extraction, SVM training and word modification.

1. feature extraction

To begin with, according to Text feature extraction, we need to mainly consider three parts in feature extraction.

1) how to count tokens

- count the frequency of words (the number of times of a word appear in a paragraph)
  - this is the default parameter, so we want to try it
- count the occurrence of words (1 for occurrence 0 otherwise)
  - this is a reasonable choice, because we know that in word modification, removing all occurrence of a word only count once, we think this may imply that we should only count the occurrence of words in feature extraction.

2) how to tokenize

- only consider 2 or more letter words
  - this is the default parameter, but this method will ignore all punctuation, including '.' in 'Mr.', so we discard this method.
- only consider 2 or more letter characters

    - we want to try this because this is very similar to default setting
- consider all characters
    - we want to try this because this is a straightforward approach after the former one failed.

3) how to normalize

- not use Tf-idf and no normalization
    - we want to try this because it looks simply
- using Tf-idf with l2 normalization
    - we want to try this because we want to reweighing high frequency words, according to Tf–idf term weighting, this is quite reasonable choice in text feature extraction

2. SVM training

The main function of SVM training was given by project specification. According to the specification, we need to consider the following parameters:

1) kernel

although 'rbf' is the default setting, only 'linear' kernel has the weights assigned to the features, which is critical in word changing step. So, we use 'linear' kernel.

2) gamma

not available in 'linear' kernel

3) C

1.0 is the default value, but other value or even a grid search is good to try

4) degree

not available in 'linear' kernel

5) coef0

not available in 'linear' kernel

3. word modification

We use the weight of words which we got from SVM training step to modify test-data.txt. Basically, we remove the most distinguished class1 words and add some most distinguished class0 words.

We can think of two different approaches:

1) static word change (remove and add a fixed number of words)

- this is a easy approach, in this method, we have another parameter called magic_number. We remove magic_number of most distinguished class1 words and then add 20-magic_number of most distinguished class0 words.
- we call this magic_number because even we have some understanding of how this number would affect our result, we still don't have a solid mathematic way to calculate what is the best magic_number
- we try magic_number=10 because we think this is the most straightforward number to begin with.

2) dynamic word change (remove and add based on the weight)

- in this approach, every time when we remove or add words, we will compare whether removing or adding will contribute more in making our paragraphs more like class0. We implement an algrithum to do this based on the weight of words which we got from SVM training step.
- We want to try this, because this seems the best approach theoretically.

Results

1. Results in feature extraction

We find that the following combination has the best performance

- count the frequency of words
- consider all characters
- using Tf-idf with l2 normalization

2. Results in SVM training

A 'linear' kernel with a fix default C has a reasonable (even may not the best) performance

3. Results in word modification

- static word change with magic_number=10 has a normal performance
- dynamic word change has a very poor performance

Further Results and Conclusions

The result in word modification is quite interesting, because theoretically, a dynamic word change method seems works better. But in fact, it works worse than a static word change with magic_number=10.

Our understanding is that because we have so many features and relatively small number of training samples. The model may end up with an over-fitting situation. This mean that our weight list is not correct, there would be lot of noise words in it. Those words have very high or very low weight, but in fact they should have around 0 weight.

If we add these words to test-data.txt, they won't contribute to make it more like class-0. Instead, if we remove more words, it would work better because the words we removed must be in the paragraph first.

This can explain why dynamic word change has a very poor performance. The reason is in fact it will add more words than remove. Also, it shows that we should remove more words, so we test different magic_number and get following result:

- static word change with magic_number=20
  - This is good to try if removing is better than adding
- static word change with magic_number=15
  - we just choose a number between 15 and 20, because we think the words have

highest weight may not be noise.

The result shows that this two magic_number have very similar performance. Because we think in the final test, professor may use a larger test set (we are not so sure), the over-fitting problem may not be so serious. Therefore, static word change with magic_number=15 may have a better performance than static word change with magic_number=20.