Naive Text Classification (bag-of-words)

Zbigniew Dendzik
Institute of Physics, University of Silesia

2025

Introduction

The goal of this exercise set is to introduce a simple example of text classification based on the bag-of-words model and a Naive Bayes classifier. The idea is to represent a document as a vector of word counts, ignoring word order. Then we train a simple statistical model that, based on these counts, tries to assign the document to one of several categories (for example “sports”, “politics”, “technology”).

In practice, similar ideas have been used for years in spam filters, simple email classification and sentiment analysis. Modern systems use much more sophisticated text representations and models (embeddings, neural networks, large language models), but bag-of-words and Naive Bayes remain a simple and surprisingly effective starting point.

In this set we will:

- implement a simple bag-of-words representation of a document,
- build a small labelled training set,
- implement a Naive Bayes classifier,
- test the classifier and compute a simple accuracy measure.

Example I – text representation

Below you will find the definition of the `TrainingDocument` class, representing a short text and its label (class).

```
1  class TrainingDocument
2  {
3     private String text;
4     private String label; //e.g. "sports", "politics"
5
6     public TrainingDocument(String text, String label)
7     {
8        this.text=text;
9        this.label=label;
10    }
11
12    public String getText()
```

```
    {
        return text;
    }

    public String getLabel()
    {
        return label;
    }
}
```

Listing 1: TrainingDocument.java

For the bag-of-words representation we use the `WordVector` class, which stores word counts in a map.

```
import java.util.*;

class WordVector
{
    private Map<String,Integer> counts;

    public WordVector()
    {
        counts=new HashMap<String,Integer>();
    }

    public void fromText(String text)
    {
        counts.clear();
        String txt=text.toLowerCase();
        txt=txt.replaceAll("[^a-z0-9 ]","");
        String[] words=txt.split("\\s+");
        for(String w:words)
        {
            if(w.length()==0) continue;
            Integer c=counts.get(w);
            if(c==null) c=0;
            counts.put(w,c+1);
        }
    }

    public int freq(String word)
    {
        Integer c=counts.get(word.toLowerCase());
        if(c==null) return 0;
        return c;
    }

    public Set<String> words()
    {
        return counts.keySet();
    }
}
```

Listing 2: WordVector.java

Ex. 2.1

Write a program `TestWordVector` that:

- reads several short texts from the keyboard,
- for each text builds a `WordVector` object,
- prints the list of words and their frequencies.

```
import java.util.*;

public class TestWordVector
{
   public static void main(String[] args)
   {
      Scanner in=new Scanner(System.in);
      System.out.println("Enter text:");
      String text=in.nextLine();

      WordVector v=new WordVector();
      v.fromText(text);

      System.out.println("Words and frequencies:");
      for(String w : v.words())
      {
         System.out.println(w+" -> "+v.freq(w));
      }
   }
}
```

Listing 3: TestWordVector.java – skeleton

Example II – training set

In this example we build a small hand-crafted training set. Each document has text and a label.

```
import java.util.*;

class TrainingSet
{
   private List<TrainingDocument> docs;

   public TrainingSet()
   {
      docs=new ArrayList<TrainingDocument>();
```

```
10      }
11
12      public void add(TrainingDocument d)
13      {
14          docs.add(d);
15      }
16
17      public List<TrainingDocument> getDocuments()
18      {
19          return docs;
20      }
21
22      public Set<String> labels()
23      {
24          Set<String> s=new HashSet<String>();
25          for(TrainingDocument d : docs)
26              s.add(d.getLabel());
27          return s;
28      }
29  }
```

Listing 4: TrainingSet.java

Ex. 2.2

Write a program `SampleData` that creates a `TrainingSet` and adds several simple examples, for example:

- a few documents in the "sports" class,
- a few documents in the "politics" class,
- a few documents in the "technology" class.

Print:

- the number of documents in each class,
- example texts from each class.

```
1   public class SampleData
2   {
3       public static TrainingSet buildSample()
4       {
5           TrainingSet set=new TrainingSet();
6
7           set.add(new TrainingDocument(
8               "The football match ended in a draw.",
9               "sports"));
10
11          set.add(new TrainingDocument(
12              "The new coach of the national team was chosen.",
```

```
            "sports"));

        set.add(new TrainingDocument(
            "The parliament debate was about a new law.",
            "politics"));

        set.add(new TrainingDocument(
            "The president gave a speech about the economy.",
            "politics"));

        set.add(new TrainingDocument(
            "The new smartphone offers a fast processor and 5G.",
            "technology"));

        set.add(new TrainingDocument(
            "The company announced a new laptop.",
            "technology"));

        return set;
    }

    public static void main(String[] args)
    {
        TrainingSet set=buildSample();

        java.util.Map<String,Integer> counter=
            new java.util.HashMap<String,Integer>();
        for(TrainingDocument d : set.getDocuments())
        {
            String label=d.getLabel();
            Integer x=counter.get(label);
            if(x==null) x=0;
            counter.put(label,x+1);
        }

        System.out.println("Documents per class:");
        for(String label : counter.keySet())
        {
            System.out.println(label+" -> "+counter.get(label));
        }
    }
}
```

Listing 5: SampleData.java – skeleton

Example III – Naive Bayes classifier

A Naive Bayes classifier assumes:

- each class has some probability distribution of words,

- words in a document are conditionally independent (naive assumption),
- for a new document we compute which class is most probable.

Our simple implementation will:

- count word frequencies in documents of each class,
- count the number of documents per class (P(class)),
- use a logarithmic version of Bayes' formula with Laplace smoothing.

```
1  import java.util.*;
2
3  class NaiveBayes
4  {
5     private TrainingSet set;
6     private Map<String,Integer> docsPerLabel;
7     private Map<String,Map<String,Integer>> wordCountsPerLabel;
8     private Map<String,Integer> totalWordsPerLabel;
9     private Set<String> vocabulary;
10
11    public NaiveBayes(TrainingSet set)
12    {
13       this.set=set;
14       docsPerLabel=new HashMap<String,Integer>();
15       wordCountsPerLabel=new HashMap<String,Map<String,Integer
            >>();
16       totalWordsPerLabel=new HashMap<String,Integer>();
17       vocabulary=new HashSet<String>();
18    }
19
20    public void train()
21    {
22       for(TrainingDocument d : set.getDocuments())
23       {
24          String label=d.getLabel();
25          Integer nd=docsPerLabel.get(label);
26          if(nd==null) nd=0;
27          docsPerLabel.put(label,nd+1);
28
29          Map<String,Integer> map=
30             wordCountsPerLabel.get(label);
31          if(map==null)
32          {
33             map=new HashMap<String,Integer>();
34             wordCountsPerLabel.put(label,map);
35          }
36
37          WordVector v=new WordVector();
38          v.fromText(d.getText());
39          for(String w : v.words())
40          {
```

```
41              vocabulary.add(w);
42              int c=v.freq(w);
43              Integer prev=map.get(w);
44              if(prev==null) prev=0;
45              map.put(w,prev+c);
46
47              Integer sum=totalWordsPerLabel.get(label);
48              if(sum==null) sum=0;
49              totalWordsPerLabel.put(label,sum+c);
50          }
51      }
52  }
53
54  public String classify(String text)
55  {
56      WordVector v=new WordVector();
57      v.fromText(text);
58
59      double bestLogP=Double.NEGATIVE_INFINITY;
60      String bestLabel=null;
61
62      int N=set.getDocuments().size();
63      int V=vocabulary.size();
64
65      for(String label : set.labels())
66      {
67          int nd=docsPerLabel.get(label);
68          double logP=Math.log((double)nd/(double)N);
69
70          Map<String,Integer> map=
71              wordCountsPerLabel.get(label);
72          int sumLabel=totalWordsPerLabel.get(label);
73
74          for(String w : v.words())
75          {
76              int tf=v.freq(w);
77              if(tf==0) continue;
78
79              Integer count=map.get(w);
80              if(count==null) count=0;
81
82              double pWord=
83                  (count + 1.0) /
84                  (sumLabel + V);
85
86              logP+=tf*Math.log(pWord);
87          }
88
89          if(logP>bestLogP)
90          {
91              bestLogP=logP;
```

```
92              bestLabel=label;
93          }
94      }
95
96      return bestLabel;
97   }
98 }
```

Listing 6: NaiveBayes.java

Ex. 2.3

Write a program `TestBayes` that:

- builds a sample training set (`SampleData.buildSample()`),
- creates a `NaiveBayes` object and calls `train()`,
- reads several new texts from the keyboard and prints the predicted label for each of them.

```
1  import java.util.*;
2
3  public class TestBayes
4  {
5     public static void main(String[] args)
6     {
7        TrainingSet set=SampleData.buildSample();
8
9        NaiveBayes nb=new NaiveBayes(set);
10       nb.train();
11
12       Scanner in=new Scanner(System.in);
13       while(true)
14       {
15          System.out.println("Enter text (empty=quit):");
16          String text=in.nextLine();
17          if(text.trim().isEmpty())
18             break;
19          String label=nb.classify(text);
20          System.out.println("Predicted label: "+label);
21       }
22    }
23 }
```

Listing 7: TestBayes.java – skeleton

Ex. 2.4

Prepare a few test texts by hand and check how often the classifier predicts the correct class (accuracy). Think about:

- what kinds of mistakes the classifier makes,
- whether the main problem is the small training set or the simplicity of the representation.

Ex. 2.5

Improve the text representation:

- add a list of stop words in the `WordVector` class,
- try to remove very rare words (occurring only once in the whole set),
- observe how these changes affect the classifier's performance.

Ex. 2.6*

Compare the simple bag-of-words + Naive Bayes approach with modern large language models:

- bag-of-words ignores word order and does not understand meaning,
- large language models can take into account context, relations between words and world knowledge.

Consider in which simple tasks such a naive classifier may still be useful (for example very basic email filtering, routing tickets to categories).