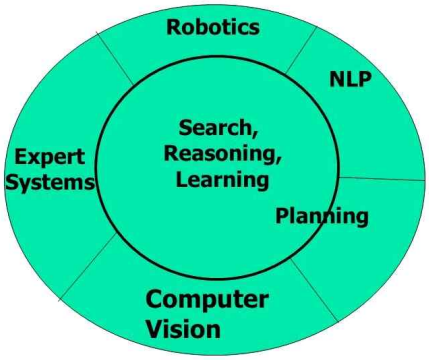
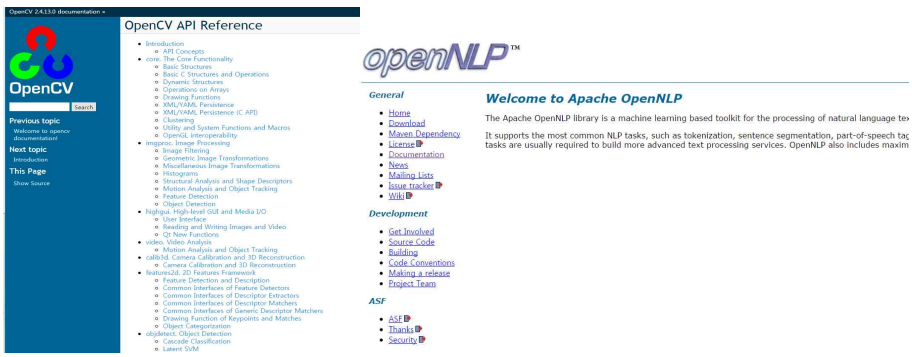
1. 개발배경 및 목적

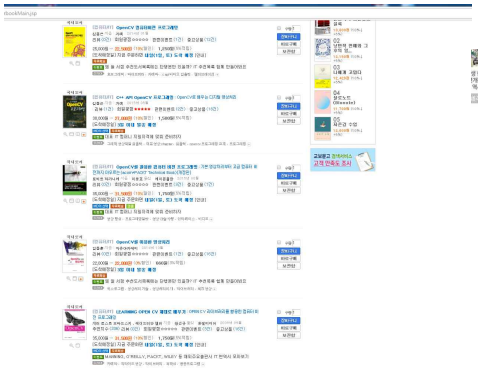
인공지능의 융용분야는 데이터 마이닝, 컴퓨터 비전와 함께 자연어 처리 역시 대표 분야로 분류되곤 한다.

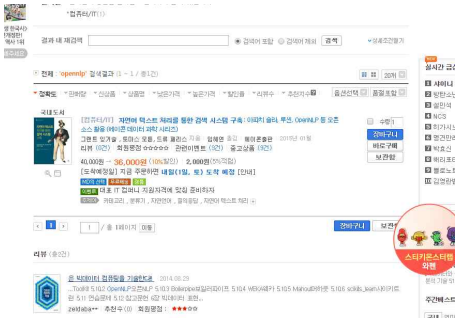


그 중 컴퓨터 비전 분야는 OpenCV에 힘입어 개발자들 사이에서 범용적으로 사용되고 있지만 자연어 처리 분야는 비교적 많은 개발자들이 생소하게 생각하는 경우가 많고 OpenCV와 같이 범용 라이브러리인 OpenNLP가 존재하지만 튜토리얼이나 관련 자료가 OpenCV에 비해서 심각하게 부족한 실정이다. 이에 본 프로젝트에서는 OpenNLP의 라이브러리 자료가 없더라도, 사용법을 모르더라도 직관적인 본 프로젝트의 라이브러리를 통해 기본적인 자연어 처리 기능을 사용 할 수 있게 하였다.



Documentation이 모두 되어있지만 꾸준한 업데이트와 자세한 예제 등에서 차이가 난다





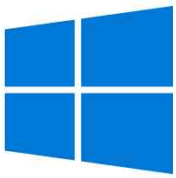
관련 도서들도 OpenCV에 비해서 종류가 거의 없다

2. 개발환경 및 개발언어

개발 환경은 Windows 계열의 OS이며, 개발 언어는 .NET Framework 4.0의 C# 환경에서 개발하였다. 라이브러리의 사용법을 예제로 나타낸 여러 데모 프로그램은 .NET Framework의 Windows Form 플랫폼에서 개발하였다.

C#

.Net Framework

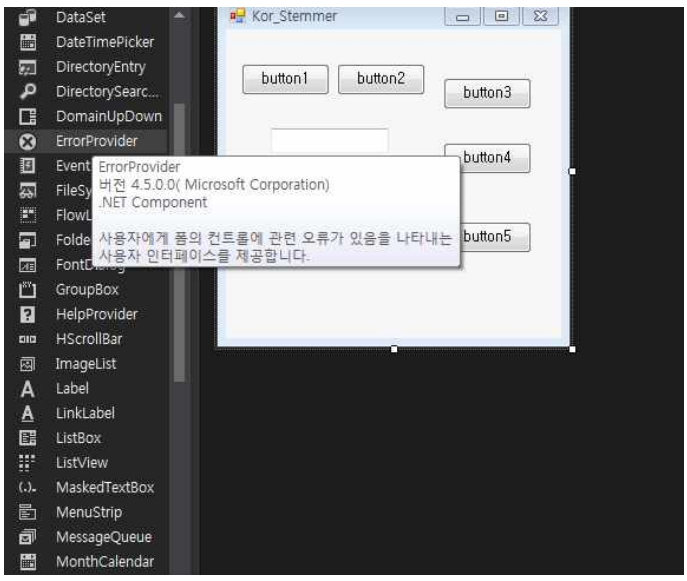


Windows OS

OOP인 C# 기반으로 개발되었기 때문에 자바로의 Converting이 용이하고 반대로 자바 계열의 OpenSource를 병합하는 것 또한 용이한 편이다(WordSentiNet, OpenNLP)

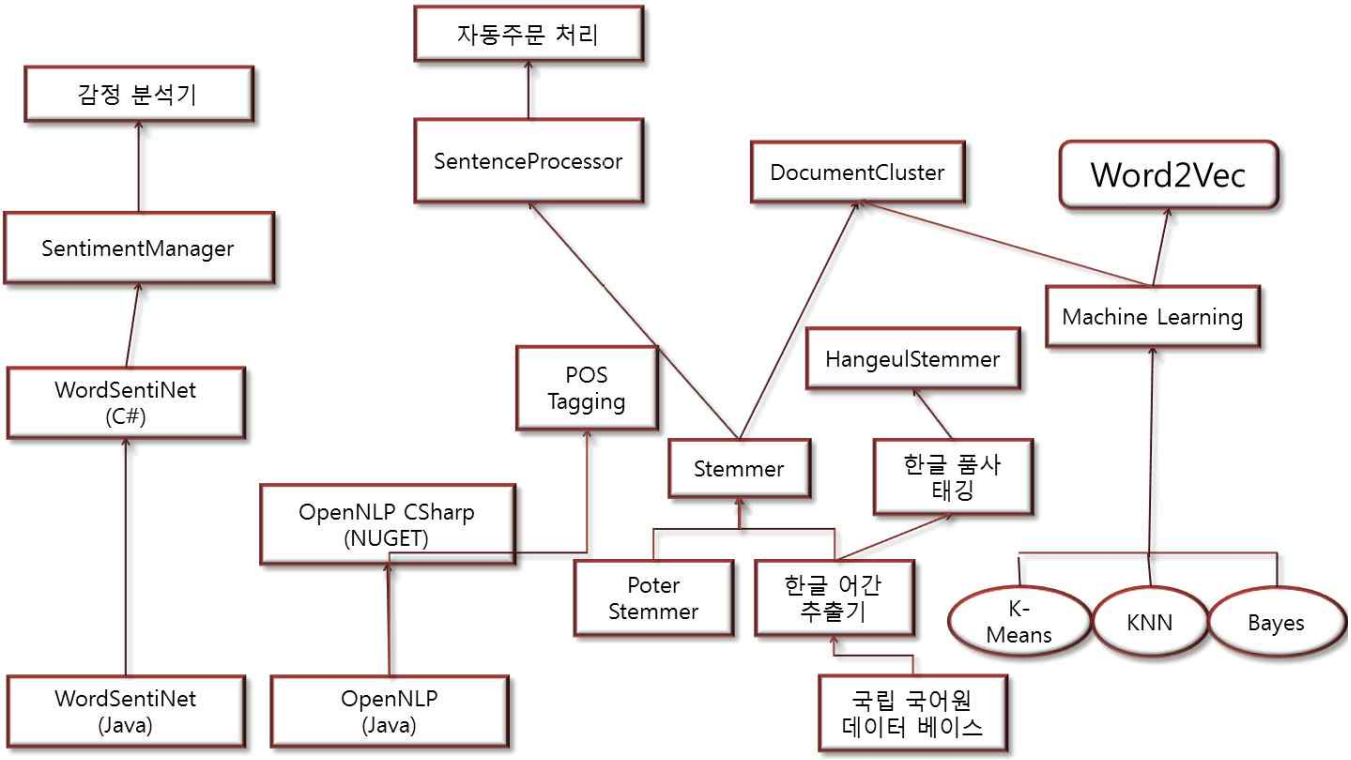
또한 C++에 관련 된 네이티브 코어 라이브러리도 DLL 등으로 익스포트 하여 사용 할 수 있는 장점이 있다.

또한 CLR(Common Language Runtime)과 CLS(Common Language Specification) 엔진은 매우 강력한 기능과 수행 능력을 가지고 있다. 또한 C#에서 사용 할 수 있는 .Net Framework는 여러 편리한 기능들을 포함하고 있다.



쉽고 편하게 UI 개발을 할 수 있는 toolbox도 .Net Framework의 기능이다

3. 시스템 구성 및 아키텍처



VELP 시스템 구성도

본 라이브러리의 시스템 구성도는 다음과 같다.

우선 WordSentiNet(Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) license)과 OpenNLP(Apache License Ver 2.0)를 가져와 사용하였고 감정 분석기 데모 프로그램은 WordSentiNet을 통해 개발한 SentimentManager 클래스를 통해 각 단어의 Positive / Negative 아웃풋을 통해 구현하였다.

Word Tree는 OpenNLP의 POS Tagging 기능을 기반으로 구현하였으며, SentenceProcessor 클래스를 통해 총 11 Step의 과정을 거쳐 Word Tree를 생성한다.

github ‘Korean Stemmer’ 검색 결과

한글 어간 추출기는 현재(2016년 9월)를 기준으로 최대 오픈소스 사이트인 git hub에는 2 ~ 3가지 프로젝트 밖에 존재하지 않았으며, 또한 심지어 한글 단어 사전의 오픈 API 또한 모두 막혀있었기 때문에, 한글의 여러 가지 많은 패턴들을 단순 조건문 코드로 추출하가는 어려웠다, 그래서 한글 어간 추출기를 직접 구현하여 라이브러리에 포함시키게 되었다.

총 1496건이 있습니다. 10개씩 보기

번호	제목	자료구분	올린사람	올린날짜	조회 수
1496	독백_여행이야기#2, 전자전사자료	외부용	관리자	2014-01-09	4275
1495	설교_교회목사#2, 전자전사자료	외부용	관리자	2014-01-09	679
1494	설교_교회목사#1, 전자전사자료	외부용	관리자	2014-01-09	477
1493	폐회사_한세추, 전자전사자료	외부용	관리자	2014-01-09	317
1492	개회사_한세추, 전자전사자료	외부용	관리자	2014-01-09	274
1491	개회사_아카데미#3, 전자전사자료	외부용	관리자	2014-01-09	251
1490	개회사_아카데미#2, 전자전사자료	외부용	관리자	2014-01-09	142
1489	개회사_아카데미#1, 전자전사자료	외부용	관리자	2014-01-09	229
1488	강연_아이발달, 전자전사자료	외부용	관리자	2014-01-09	495
1487	강연_골다공증, 전자전사자료	외부용	관리자	2014-01-09	294

맨 처음 이전 1 2 3 4 5 6 7 8 9 10 다음 맨 마지막

한국 국립 국어원 말뭉치 공개 데이터 베이스

그 외에도 머신 러닝 알고리즘들(KNN, K-Means Clustering, Bayesian Classifier, HMM)을 활용하여 문서 분류, Language Model의 일종인 Word2Vec과 HHM으로 구현한 Language Model등이 프로젝트.MachineLearning namespace에 포함되어 있다.

4. 프로젝트 주요기능

이 툴에는 기본적으로 NLP의 사용외에도 자체적인 Laguage Tree 생성 알고리즘을 포함하고 있다. 한국의 영어 교육에서는 제일 기본적으로 영어의 문장 형식을 5가지로 나누어 교육을 하고 있다.

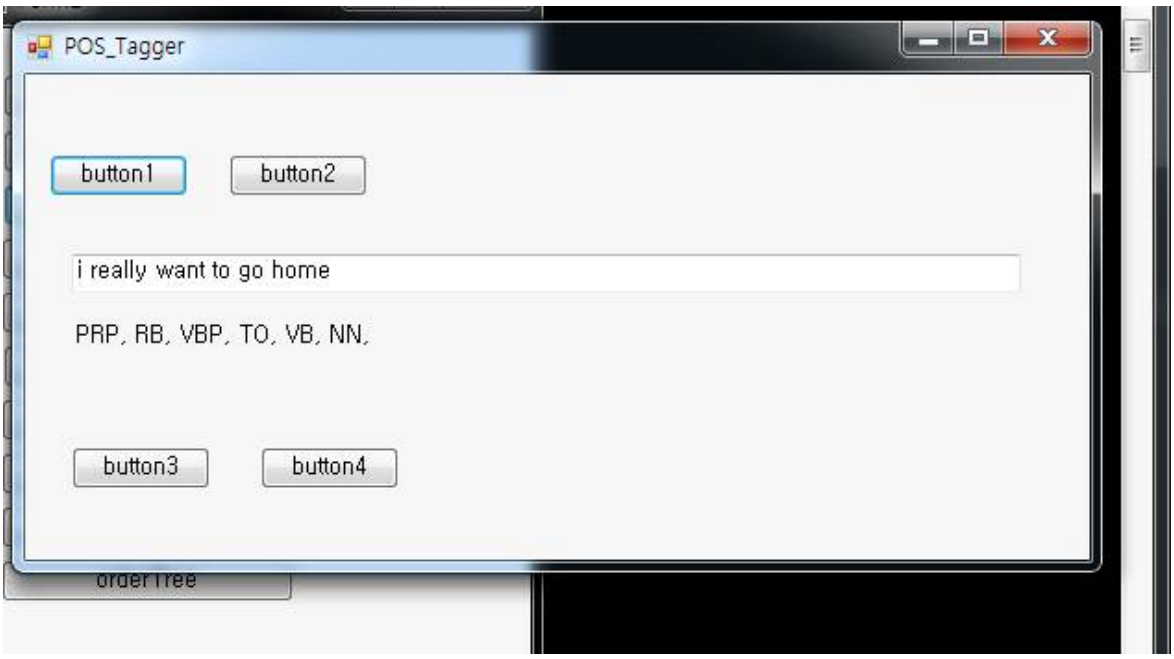
1	s	v		
2	s	v	c	
3	s	v	o	
4	s	v	o	o
5	s	v	o	c

표1 영어의 문장 형식 체계

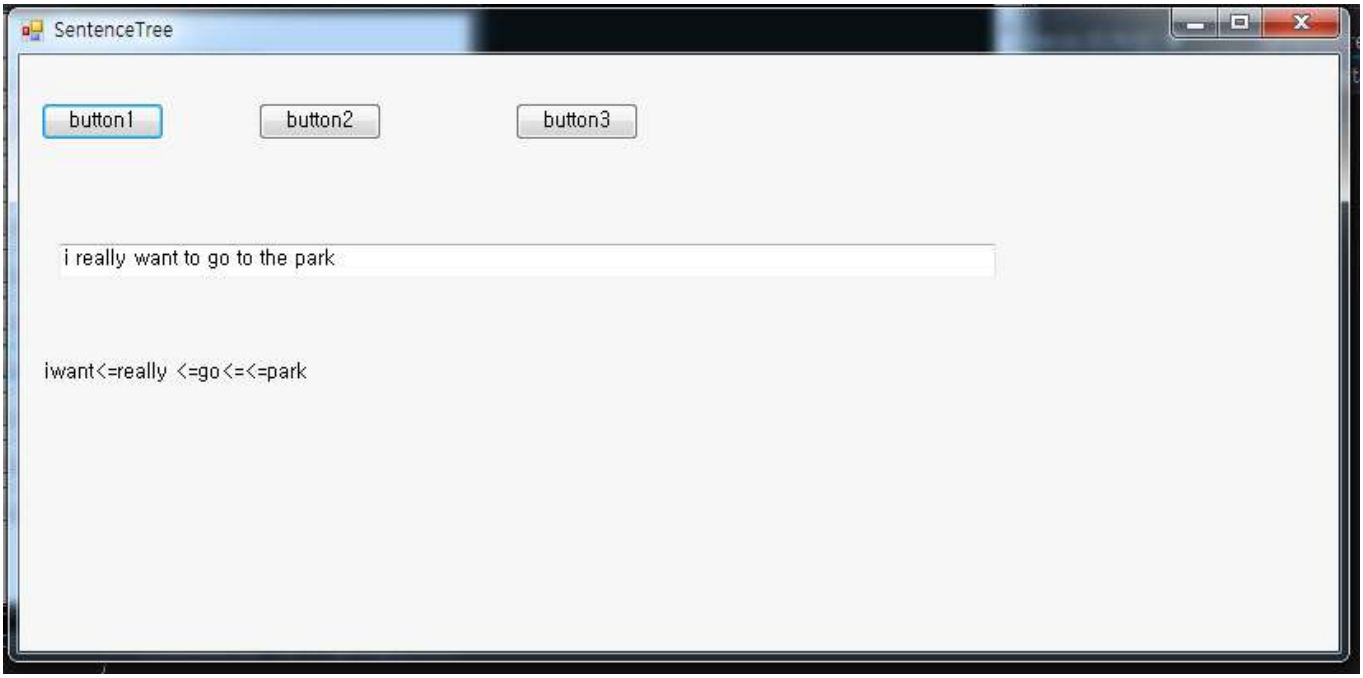
표1의 주어, 동사, 형용사, 보어들을 통해 문장 형식을 매기는 것은 매우 쉽지만 영어의 언어에는 여러 가지 부가적인 요소들이 들어가게 된다. (부사, 관계대명사, 동명사, 전치사 구문 등...)

VELP의 SentenceProcessor 클래스에는 S, V, O, C 등과 같이 문장의 형식에 결정을 주는 역할이 아닌 다른 꾸며주는 단어들을 모두 트리에 추가시키고 삭제하여 문장의 형식을 쉽게 결정 할 수 있는 기능을 제공한다. 결과를 얻기 위한 매개변수는 NLP의 POS Tagging을 통해 얻는 단어와 결과를 얻으려는 문장을 넣어주면 된다.

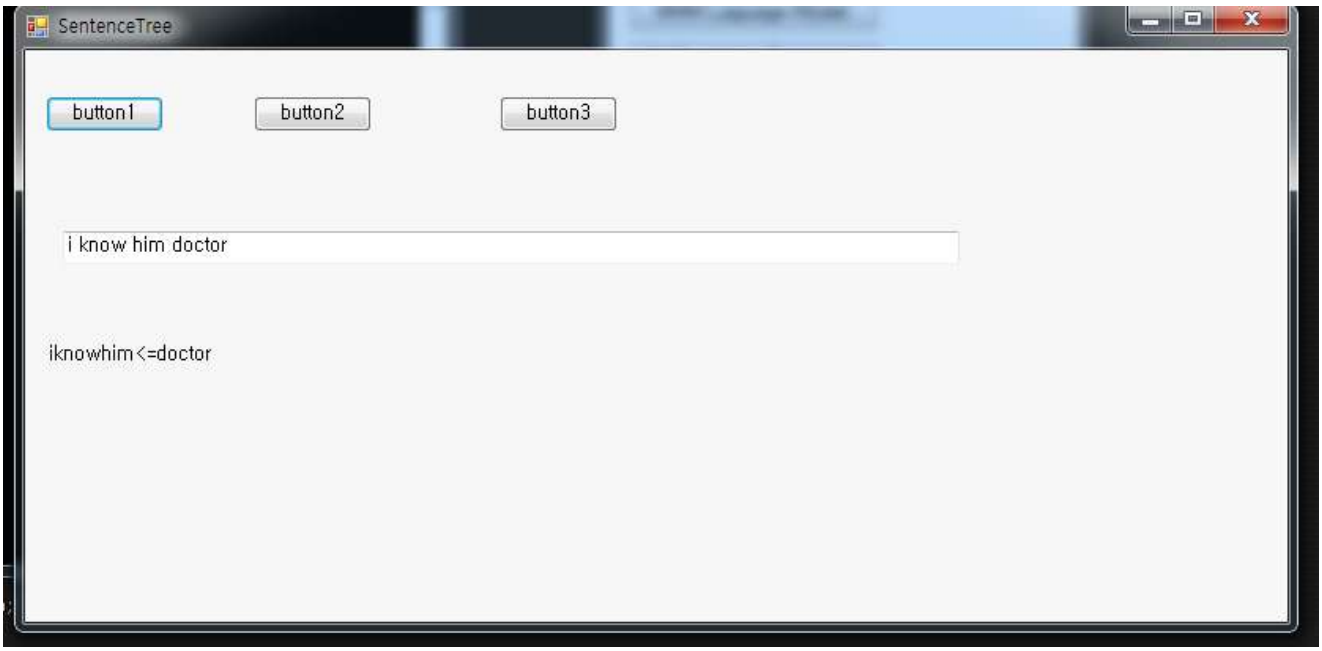
POS Tagging은 Penn Tree Bank를 이용하고 있는 OpenNLP를 이용했다.



Pos Tagging 결과

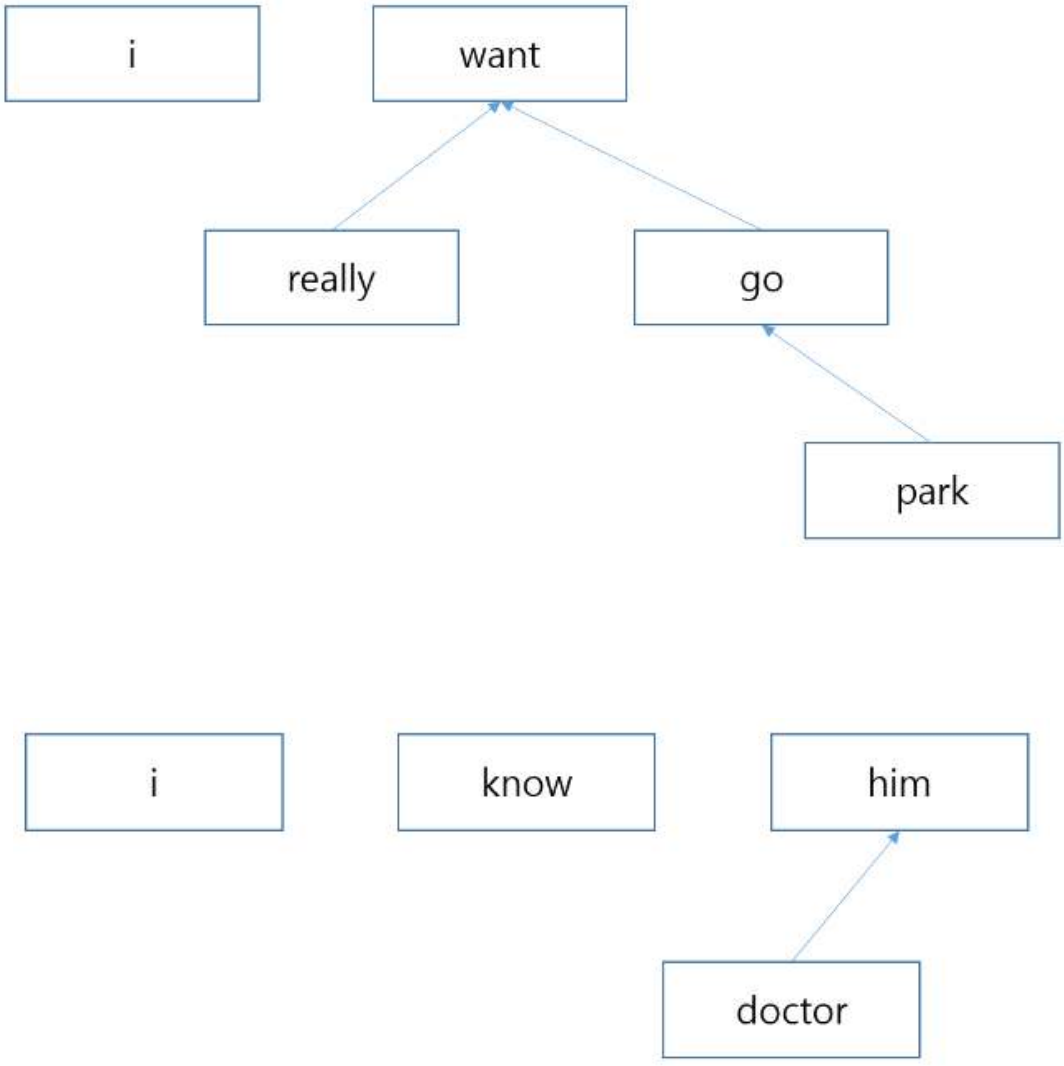


SentenceProcessor 클래스를 통해 결과를 적용 시킨 결과 예시1



SentenceProcessor 클래스를 통해 결과를 적용 시킨 결과 예시2

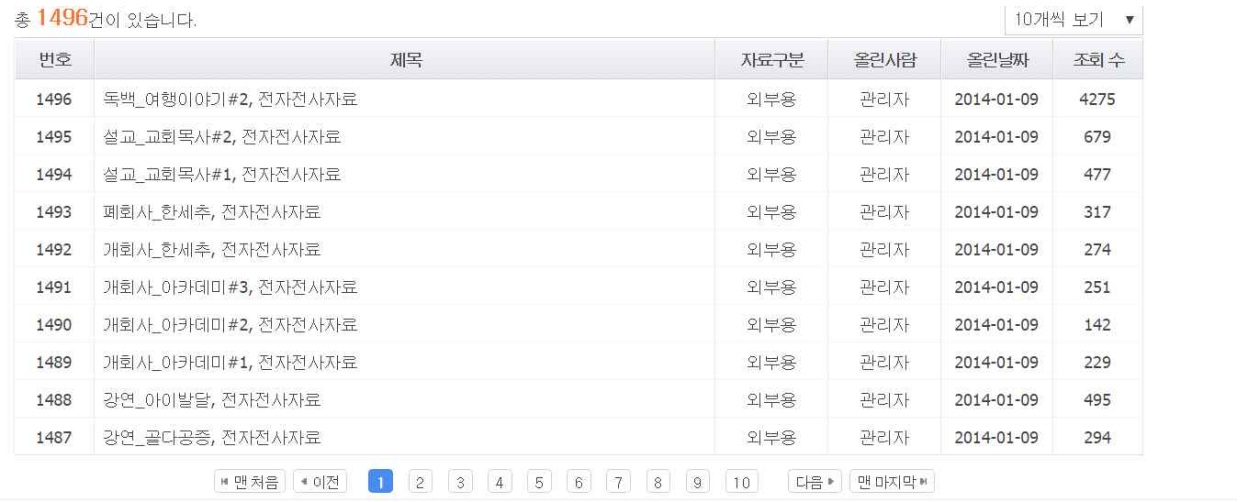
예시 1의 문장을 살펴보면 "i really want to go to the park"에서
really, to go(to 부정사 목적 구문), to park(to 목적어 구문), the(관사)와 같은 꾸미는 문장이 주문장에서 제거가 되고 1형식 문장에 필요한 주어(i)와 동사(want)만 남은 결과를 볼 수 있다.



예시 1,2의 결과를 Language Tree로 나타낸 그림

English Lauguage Processing외에도 한글에 관련 된 기능도 포함되어 있는데 Hanguel Stemmer 클래스에서 한글 어간 추출과 품사 태깅을 사용 할 수 있다.

한글에는 영어의 PoterStemmer와 다르게 복잡한 규칙들이 있어 Porter Stemmer처럼 단순한 패턴을 통한 어간 추출은 힘들었다. 그래서 POS Tagging과 같이 사람에 의해 Tagging 된 데이터를 통해 자동 어간 추출기를 구현했다.

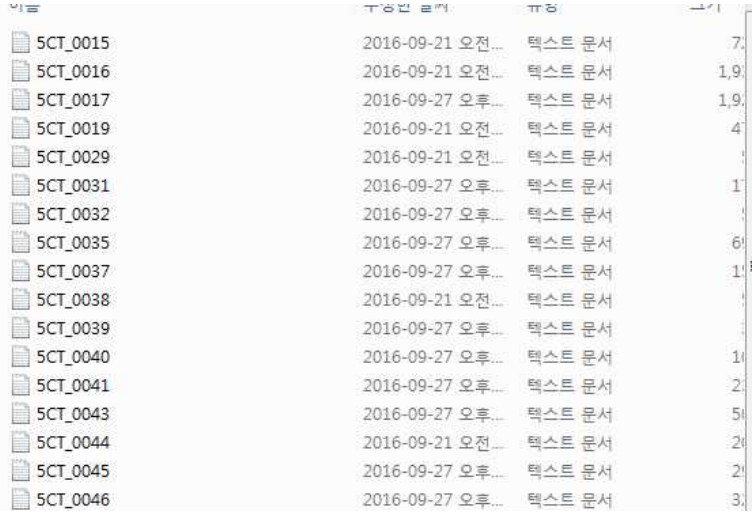

총 1496건이 있습니다. 10개씩 보기

번호	제목	자료구분	올린사람	올린날짜	조회 수
1496	독백_여행이야기#2, 전자전사자료	외부용	관리자	2014-01-09	4275
1495	설교_교회목사#2, 전자전사자료	외부용	관리자	2014-01-09	679
1494	설교_교회목사#1, 전자전사자료	외부용	관리자	2014-01-09	477
1493	폐회사_한세추, 전자전사자료	외부용	관리자	2014-01-09	317
1492	개회사_한세추, 전자전사자료	외부용	관리자	2014-01-09	274
1491	개회사_아카데미#3, 전자전사자료	외부용	관리자	2014-01-09	251
1490	개회사_아카데미#2, 전자전사자료	외부용	관리자	2014-01-09	142
1489	개회사_아카데미#1, 전자전사자료	외부용	관리자	2014-01-09	229
1488	강연_아이발달, 전자전사자료	외부용	관리자	2014-01-09	495
1487	강연_골다공증, 전자전사자료	외부용	관리자	2014-01-09	294

맨 처음 이전 1 2 3 4 5 6 7 8 9 10 다음 맨 마지막

국립 국어원에서 제공하고 있는 말뭉치 데이터베이스

수 백개의 게시물에 여러 가지 다양한 주제의 말뭉치 자료들을 포함하고 있으며, 일정한 Form을 이루고있기 때문에 Parsing하여 사용 할 수 있다.


5CT_0015	2016-09-21 오전...	텍스트 문서	7
5CT_0016	2016-09-21 오전...	텍스트 문서	1,9
5CT_0017	2016-09-27 오후...	텍스트 문서	1,9
5CT_0019	2016-09-21 오전...	텍스트 문서	4
5CT_0029	2016-09-21 오전...	텍스트 문서	
5CT_0031	2016-09-27 오후...	텍스트 문서	1
5CT_0032	2016-09-27 오후...	텍스트 문서	
5CT_0035	2016-09-27 오후...	텍스트 문서	6
5CT_0037	2016-09-27 오후...	텍스트 문서	1
5CT_0038	2016-09-21 오전...	텍스트 문서	
5CT_0039	2016-09-27 오후...	텍스트 문서	
5CT_0040	2016-09-27 오후...	텍스트 문서	1
5CT_0041	2016-09-27 오후...	텍스트 문서	2
5CT_0043	2016-09-27 오후...	텍스트 문서	5
5CT_0044	2016-09-21 오전...	텍스트 문서	2
5CT_0045	2016-09-27 오후...	텍스트 문서	2
5CT_0046	2016-09-27 오후...	텍스트 문서	3

본 프로젝트에서 사용한 데이터 파일 목록들

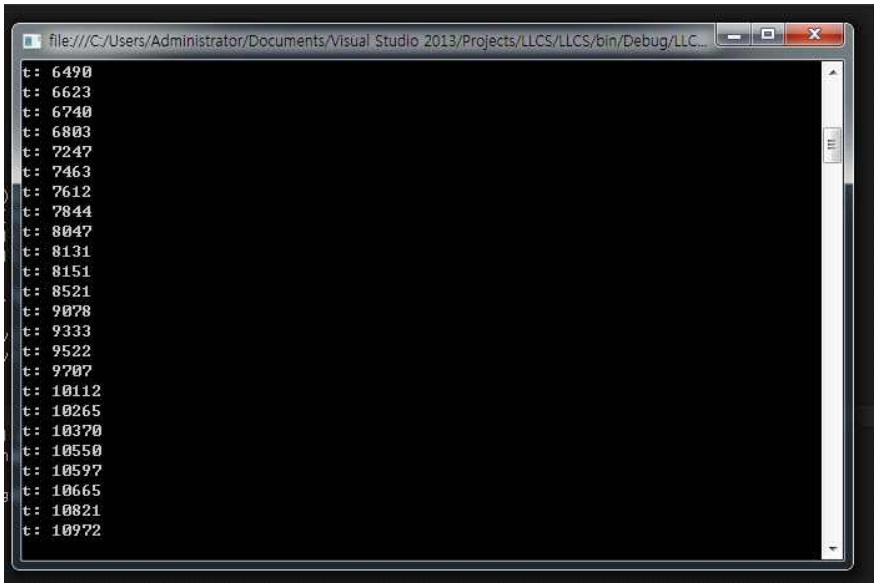
```
5CT_0015-0000070        </s>
5CT_0015-0000080        </u>
5CT_0015-0000090        <u who="P2">
5CT_0015-0000100        <s n="00002">
5CT_0015-0000110        많이    많이/MAG
5CT_0015-0000120        가      가/VV+ㅏ/E
5CT_0015-0000130        봤지.   보/VX+ㅏㅆ
5CT_0015-0000140        </s>
5CT_0015-0000150        <s n="00003">
5CT_0015-0000160        근데,   근데/MAJ+,
5CT_0015-0000170        </s>
5CT_0015-0000180        </u>
5CT_0015-0000190        <u who="P1">
5CT_0015-0000200        <s n="00004">
5CT_0015-0000210        예.     예/IC+./SF
5CT_0015-0000220        </s>
5CT_0015-0000230        </u>
5CT_0015-0000240        <u who="P2">
5CT_0015-0000250        <s n="00005">
5CT_0015-0000260        옷은    옷/NNG+은/
5CT_0015-0000270        많이    많이/MAG
5CT_0015-0000280        안      안/MAG
5CT_0015-0000290        샀어    사/VV+ㅏㅆ
5CT_0015-0000300        나는,   나/NP+는/J
5CT_0015-0000310        </s>
```

파일 내부의 모양은 다음과 같다

왼 쪽의 단어는 어간을 추출 할 단어이고 오른쪽은 추출 된 단어이며 그 옆에는 단어의 품사가 붙어있다.

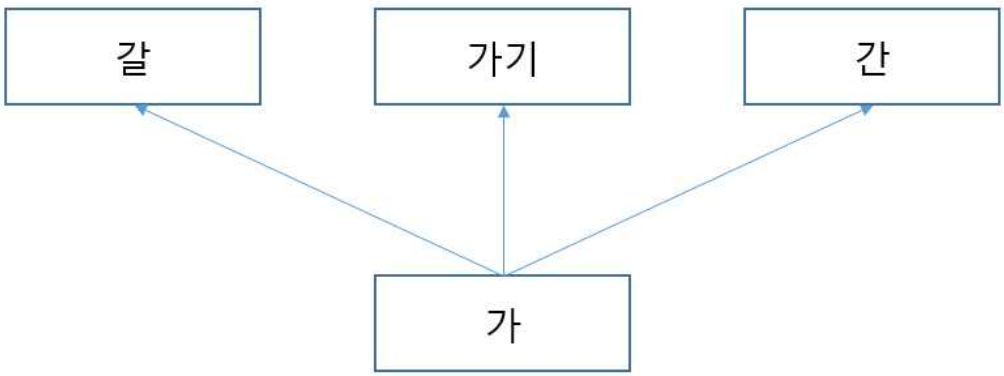
```
string[] token = str.Split('₩t');

if (token.Length == 3)
{
    char[] posCh = new char[2];

    string word = token[1];
    string key = token[2].Split('/')[0];
    string temp = key.ToCharArray()[0] + "";
    posCh[0] = token[2].Split('/')[1].ToCharArray()[0];
    posCh[1] = token[2].Split('/')[1].ToCharArray()[1];

    string firstChar = Seperate(key.ToCharArray()[0] + "");

    char vow = firstChar.ToCharArray()[1];
    char con = firstChar.ToCharArray()[2];

    char[] chs = word.ToCharArray();

    if(chs[chs.Length - 1] == ',' || chs[chs.Length - 1] == '.'){
        char[] nw = new char[chs.Length - 1];

        for (int a = 0; a < chs.Length - 1; a++)
        {
            nw[a] = chs[a];
        }

        word = new string(nw);
    }
```

Parsing 부분의 소스는 다음과 같다



콘솔의 숫자는
데이터 파일을 통해 생성 된 Dictionary 안의 단어의 수를 나타내며
저 단어는 Key의 숫자를 나타낸다

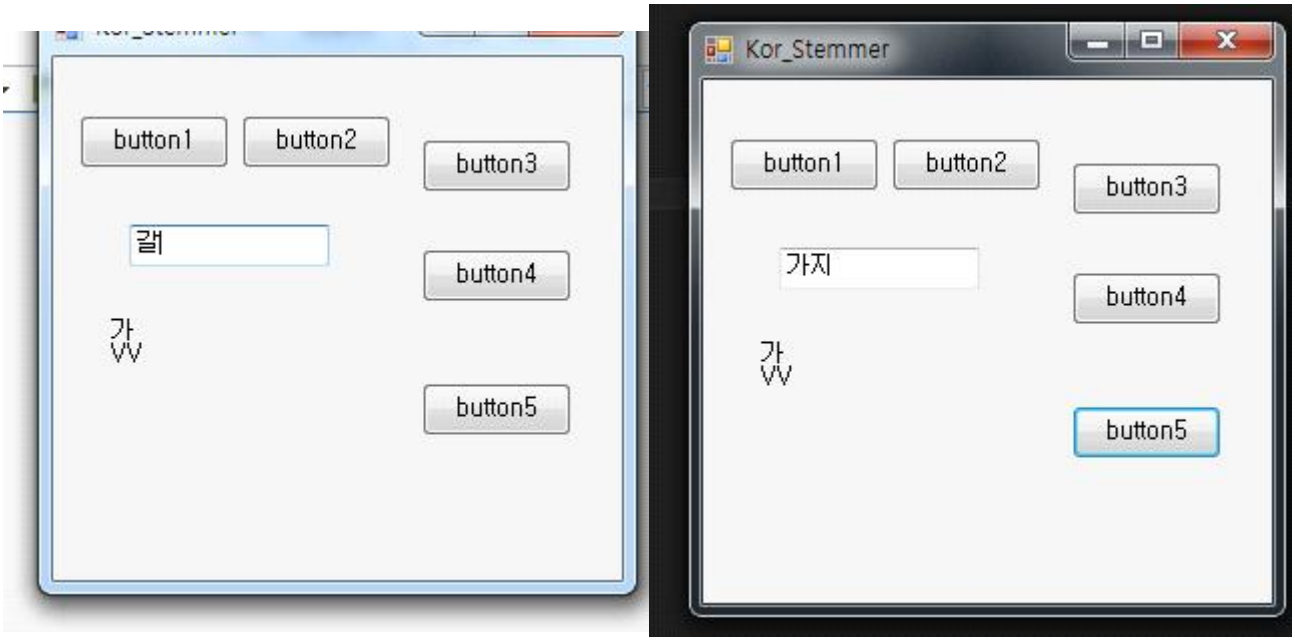
예를 들어 간, 갈, 가기 등의 단어에서 어간을 추출한다면 공통적으로 '가'가 될 것이다.

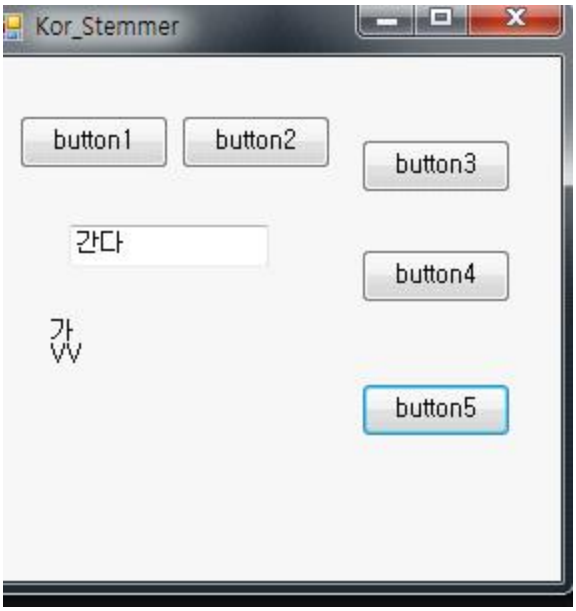


그 때 '가'는 Class에서 Key 변수에 해당하며, '갈', '가기', '간' 등은 Word 변수에 해당한다. 그렇기 때문에 Key는 국어원 데이터에 의해 10000개 정도가 생성되어 Linear Search를 하더라도 큰 무리가 없겠지만 하나의 Key에는 평균적으로 10개 정도의 Word가 붙게 되고 Linear Seach를 하게 되면 총 10만번의 연산이 필요하게 된다.

이에 클래스에서는 좀더 빠른 검색방법을 위해 600개의 Dictionary로 나누는 방법을 선택했다. 그 600개의 기준은 첫 단어의 받침을 제외한 모음 자음인데, 예를 들어 Dictionary에서 검색 할 단어가 '덜' 이라면

[(ㄷ에 해당하는 index X 총 모음의 수) + ㅓ에 해당하는 인덱스]가 나눈 Dictionary의 인덱스가 된다.





어간 '가'에 대한 여러 가지 실행 결과

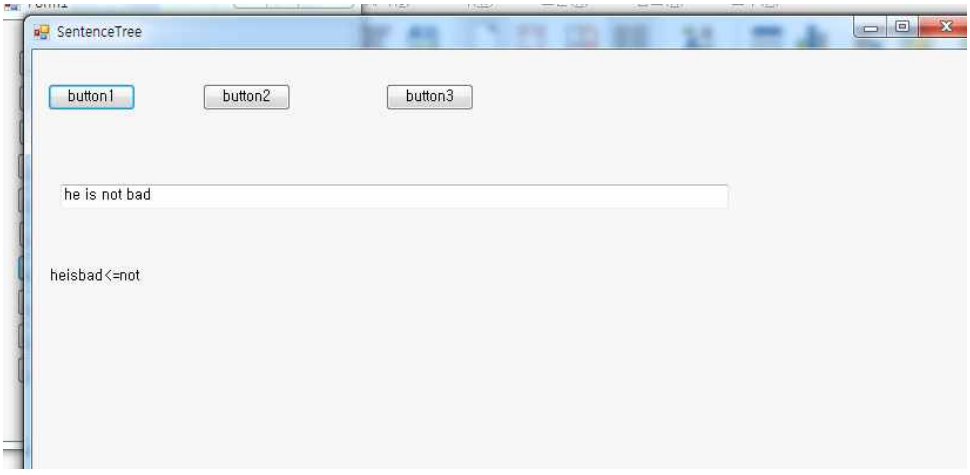
다음 라이브러리에 포함 된 기능은 Sentimental Analysis이다.

90년도에 나왔던 아이디어인, 영화 평점 사이트의 데이터를 통해 단어의 Positive Negative를 분석하는 "Thumbs up? Sentiment Classification using Machine Learning"처럼 단어의 P/N을 분석하는 방법과 오픈소스에는 여러 가지가 존재하지만 가장 접근성이 용이하면서 느슨한 라이센스로 자유로운 수정과 사용이 가능하고, 사용법이 간단한 WordSentiNet을 사용하여 Sentimental Analysis 기능을 구축했다. 기존의 오픈소스는 Java로 사용 된 소스코드를 제공했다.

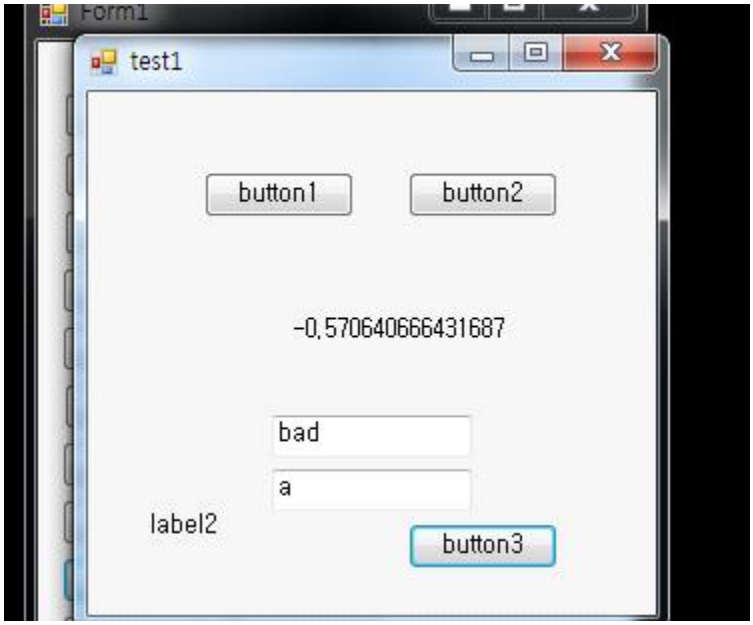
```
28
29 class SentiWordNetDemoCode {
30     private String pathToSWN = "C:\\Users\\MyName\\Desktop\\SentiWordNet_3.0.0.tx
31
32     private Map<String, Double> dictionary;
33
34     public SentiWordNetDemoCode(String pathToSWN) throws IOException {
35         // This is our main dictionary representation
36         dictionary = new HashMap<String, Double>();
37
38         // From String to list of doubles.
```

SentiWordNet의 자바 소스코드 일부분

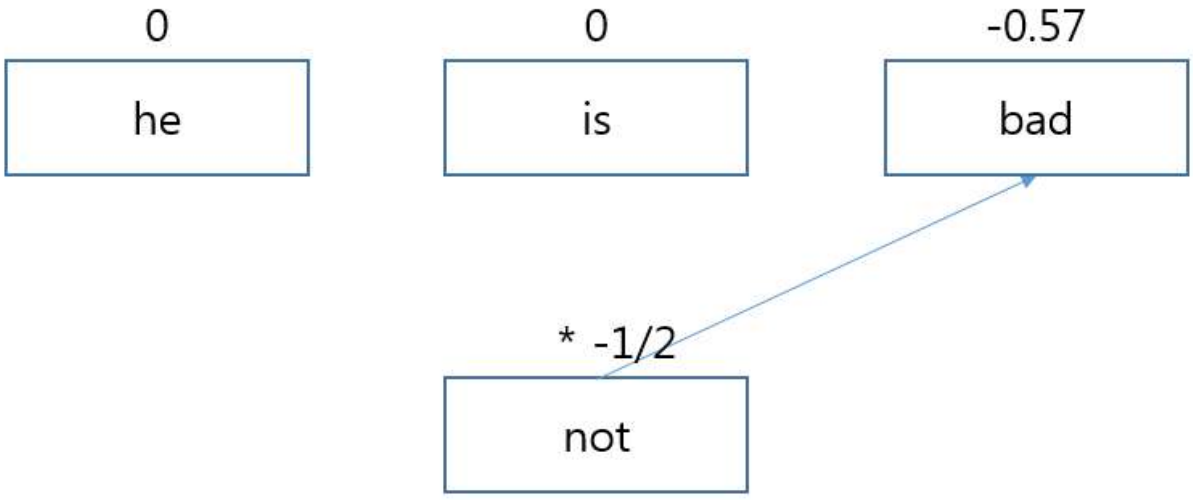
위 코드에서는 Map이란 클래스를 사용하여 단어의 P/N Value를 편리하게 검색하였는데 위 소스를 옮기기 위해 HashTable 등의 여러 가지 클래스를 사용해보았으나, 호환성이 맞지 않는 클래스들도 있었고, 더군다나 자바의 Map에 비해 검색속도가 현저하게 느려 사용이 불가능 한 클래스도 있었다. 그래서 기존의 Java코드를 조금 수정하여 Model Data 파일을 추출하도록 했고, 그 파일을 다시 본 라이브러리에서 읽어 들여 단순 배열 탐색을 통해 P/N Value를 검색하도록 하였다. 단순히 단어들의 P/N Value도 검색해볼 수 있지만 SentenceProcessor 클래스를 통해 그 문장의 P/N Value를 더 정확하게 얻을 수 있다.



he is not bad 문장의 Lagnuage Tree



bad 라는 단어를 검색 했을 때 나오는 P/N Value이다.
bad는 단어, a는 검색 할 단어의 POS(품사)를 나타낸다.

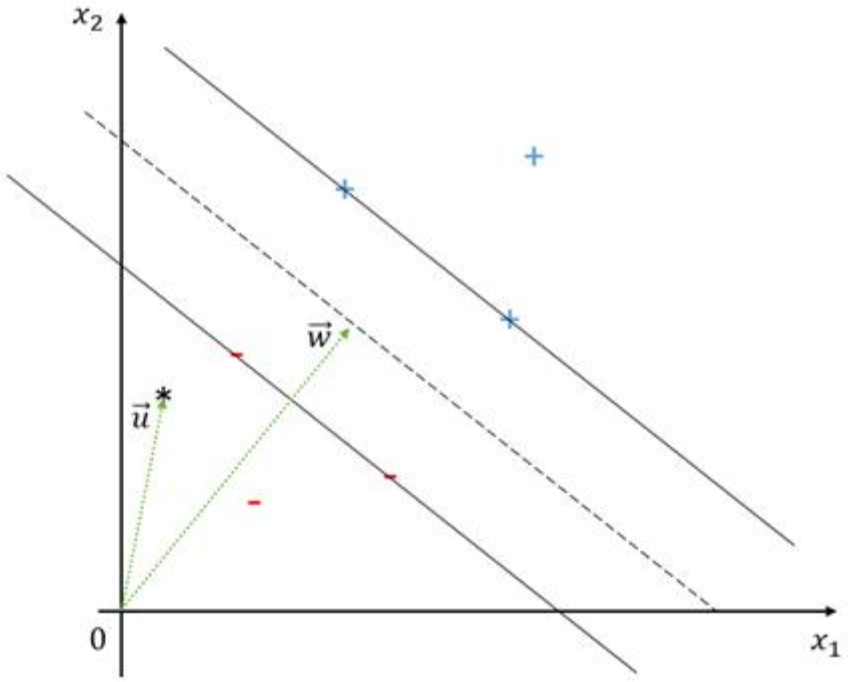


Language Tree의 모양에 각 단어의 P/N Value의 모양을 적용 시킨 결과를 적용시키면 총 – 0.29의 값이 나오는 것을 볼 수 있다.

Document Categorization

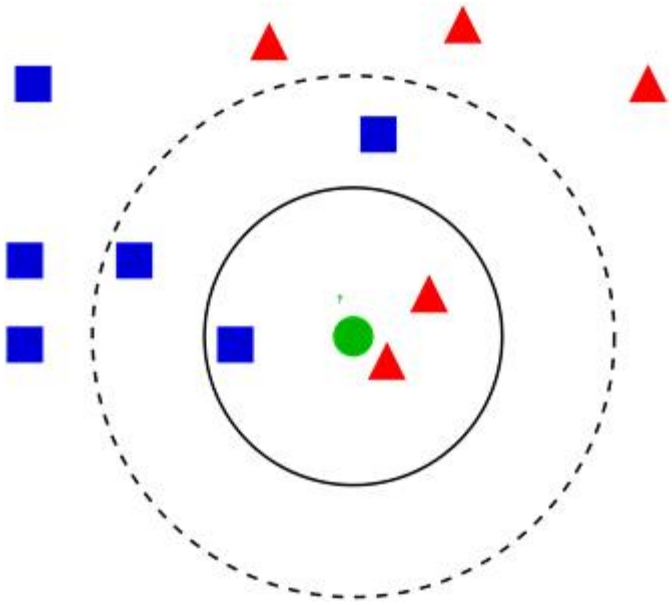
Document Categorization이란 컴퓨터 분야에서 자동으로 문서에 하나 또는 그 이상의 라벨 혹은 카테고리를 할당하는 작업을 의미한다.

Document Categorization에 많이 사용되는 메소드로는 Bayesian Calssifier, SVM(Support Vector Machine), KNN 등이 있다.



Support Vector Machine

SVM은 Support Vector를 찾아내고 그 벡터들간의 Margin을 최대화 하는 알고리즘이며, 가장 범용적으로 사용되고 있다.



K-Nearest Neighborhood

KNN 은 각 Point 간의 Distance를 계산하여 가장 가까운 K 개의 포인트의 라벨로 분류하는 알고리즘이다.

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B)} \propto \mathcal{L}(A|B)\Pr(A)$$

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i|C_k).$$

Bayesian Classifier

Bayesian Classifier는 Naive한 Bayes 확률을 계산하여 가장 큰 확률로 분류하는 알고리즘이다.

본 예제에서는 3가지 알고리즘들 중에서 Bayesian Classifier를 통해 분류기를 구현하였다.

Bayes 확률을 구하는 Feature로는 그 Document에서 등장하는 단어들의 빈도수를 사용하였다. 단어의 빈도수 계산에는 WordBag 객체를 만들어 계산하였다.

다음은 WordBag 클래스에서 빈도수가 계산되어지는 과정이다

```
public string[] getBag(string[] Str)
{
    string[] bags = Str[0].Split(); Cnts = new int[bags.Length];
    string[] resultBags = new string[50];
    //NaiveBayes에 사용할 가장 빈도가 높은 단어들입니다. 데이터의 규모가 크다면 더 큰 배열을 사용 하시는게 좋습니다.

    int numberOfBags = bags.Length;

    for (int i = 0; i < numberOfBags; i++)
    {
        Cnts[i] = 1;
    }

    for (int i = 0; i < Str.Length; i++)
    {
        string[] mybag = Str[i].Split();

        for (int j = 0; j < mybag.Length; j++)
```

WordBag 클래스

병합 된 string을 WordBag 클래스의 getBag 함수를 통해 빈도수를 구할 수 있다

$$P(C_i|D) = \frac{P(D|C_i) * P(C_i)}{P(D)} \quad (1)$$

$$\arg max(P(e_n|C_i)) \quad (2)$$

베이즈 확률을 수식으로 나타내보면 식1과 같지만 Document Categorization에서는 분류를 해야 하기 때문에 수식2를 계산하였다.
하지만 이 식을 그대로 사용하면 큰 문제가 발생하게 되는데, 만약 Test Data로 쓸려는 문서에 Training Data를 넣어서 만든 Word Bag에 없는 단어가 있다고 가정해보자. 그러면 빈도수는 0이기 때문에 단 하나의 단어라도 Wordbag에 존재하지 않으면 모두 0이 되어버리는 문제가 발생한다.
Bayesian Classifeir에서는 이러한 문제를 Laplace Smoothing, Log를 사용한 조건부 확률 계산 등을 통해 해결 하고 있는데 본 예제에서는 Log를 사용한 조건부 계산을 이용하였다.

$$\arg max(\log(P(e_n|C_i))) \quad (3)$$

조건부 계산으로 식 2를 수정하면 식 3과 같이 나오게 되는데 이 경우 WordBag에 단어가 존재하지 않아 빈도수가 1이더라도 Log(0)의 결과는 1이기 때문에 전체 확률이 0이 되는 것을 막을 수 있다.

```
for (int i = 0; i < wordbag.bags.Length; i++)
{

    for (int a = 0; a < number; a++)
    {
        if (wb[a].LinearSearch(wordbag.bags[i]) != -1)
        {
            int index = wb[a].LinearSearch(wordbag.bags[i]);
            score[a] += wb[a].Cnts[index];
        }
    }

}
```

TextCategorizer 클래스
wordbag 변수는 Test 할 문서의 Wordbag 변수이며, number는 분류 할 라벨의 개수를 의미하며, wb 변수는 분류 할 카테고리들의 단어 빈도수를 가지고 있는 WordBag 배열이다.

Document Categorization을 할 때 별로 도움이 되지 않는 단어들이 있다. 예를 들면
“이다, 있다 인데, 그런데, 하지만” 등과 같이 Category의 특징을 띄지 않고 여러 Document에서 공통적으로 등장하는 단어들이다. 이런 단어들의 중요도를 수치로 계산하는 방법이 있는데 “TF-IDF” 방법이다.
TF-IDF 는 TF(Term Frequency)와 IDF(Inverse Document Frequency)를 합친 단어이다. TF는 말그대

로 단순한 빈도수를 나타내고 IDF는 그 단어가 얼마나 많은 정보를 제공하는 가를 나타내는 지표이다.

$$\mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}} \quad (4)$$

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \quad (5)$$

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D) \quad (6)$$

TF, IDF와 TFIDF는 각각 (4), (5), (6)을 통해 계산한다.

```
참조 2개
public void get_TD_IDF( WordBag[] otherbags )
{
    TD_IDF = new double[bags.Length];

    for (int k = 0; k < bags.Length; k++ )
    {
        double tf, idf;

        int countOfWordDocuments = 1;
        int totalFrequency = Cnts[k];

        for (int i = 0; i < otherbags.Length; i++ )
        {
            int index = otherbags[i].BinarySearchWord(bags[k]);

            if( index != -1 ){
                countOfWordDocuments++;
                totalFrequency += otherbags[i].Cnts[index];
            }
        }

        try
        {
            tf = Cnts[k] / totalFrequency;
            idf = (otherbags.Length + 1) / countOfWordDocuments; idf = Math.Log(idf);

            TD_IDF[k] = tf * idf;
        }
        catch (DivideByZeroException e)
        {

        }
```

WordBag 클래스

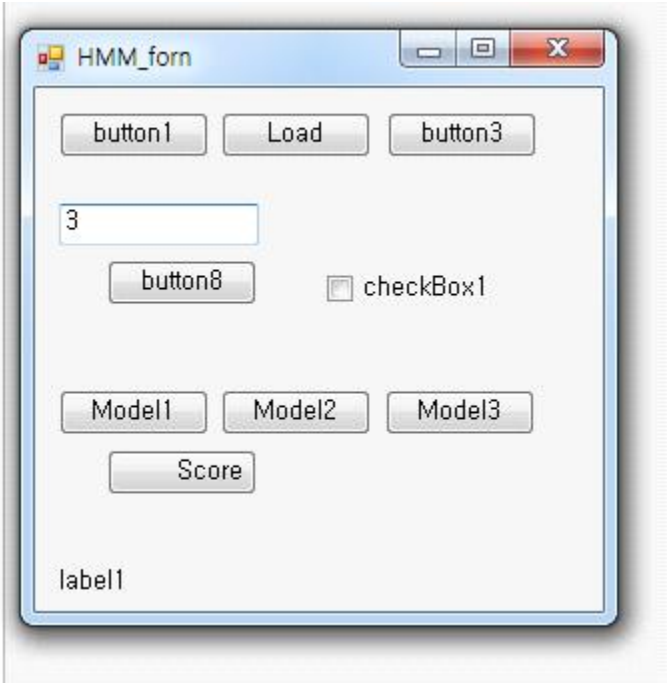
WordBag 클래스의 내장 함수인 get_TD-IDF 함수를 통해 값을 구하게 되며 매개변수는 모든 라벨의 WordBag 변수들을 배열로 묶어서 넣어주면 된다

만약 전체 빈도수가 100, 100, 100인 3개의 카테고리로 문서들을 분류한다고 가정 했을 때, “였다“라는 단어가 3개의 Category에 모두 존재하고, 각각 빈도수가 7, 10, 15 각각 TF는 0.07, 0.1, 0.15가 된다. IDF는

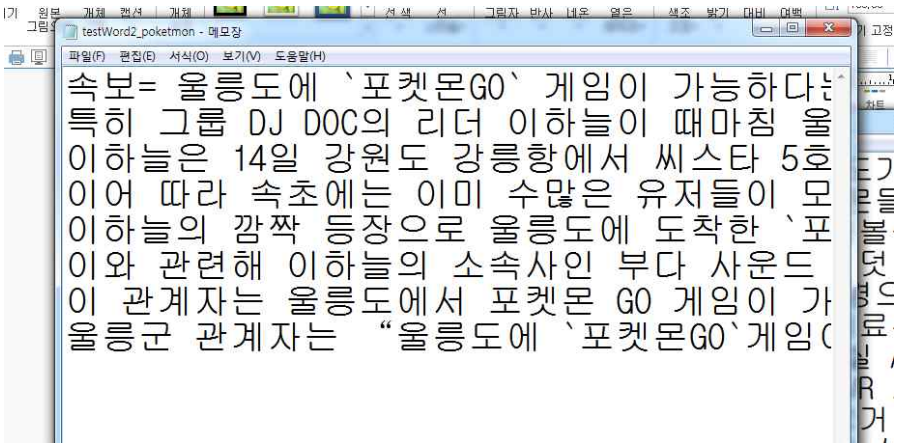
log (((3 + 1) / 4))이며 log 1은 0이기 때문에 최종 값은 0이 된다. 최종 tfidf는 0이 나오게 되는데, 모든 Document에서 등장하는 별로 중요하지 않는 단어이므로 TFIDF 값에 의해서 중요도가 낮게 나온 것을 알 수 있다.

$$\arg max(\log(\ P(e_n|C_i)) * tf - idf_i\ ) \quad (7)$$

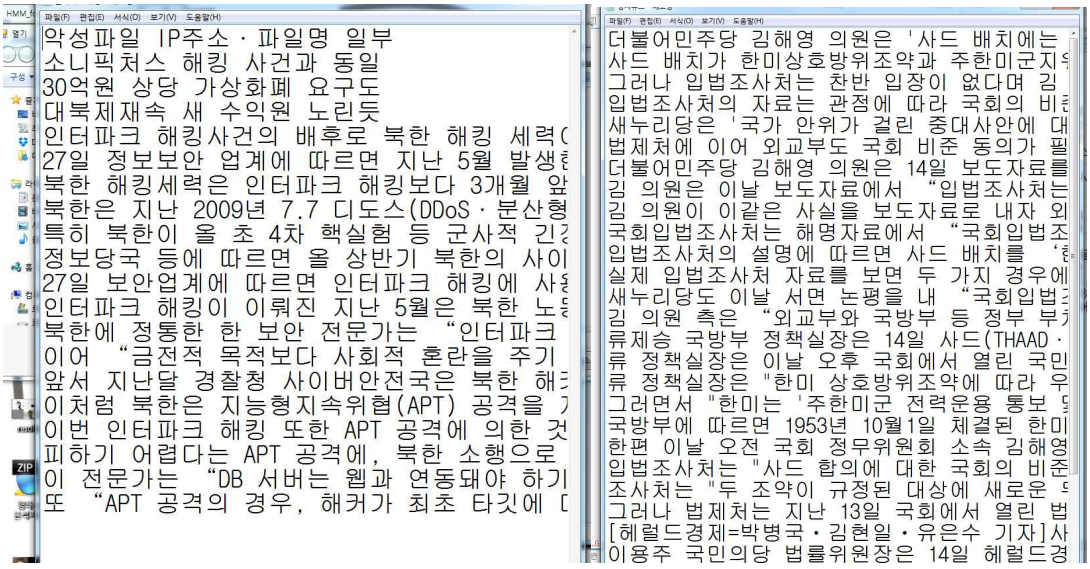
Document를 분류하기 위해 구하는 Bayes 확률의 최종 수식은 7과 같이 나오게 된다.



TextBox 안의 숫자는 라벨의 숫자이다. 왜냐하면 TextCategorizer 클래스를 초기화 할 때 그 숫자가 필요하기 때문이다.



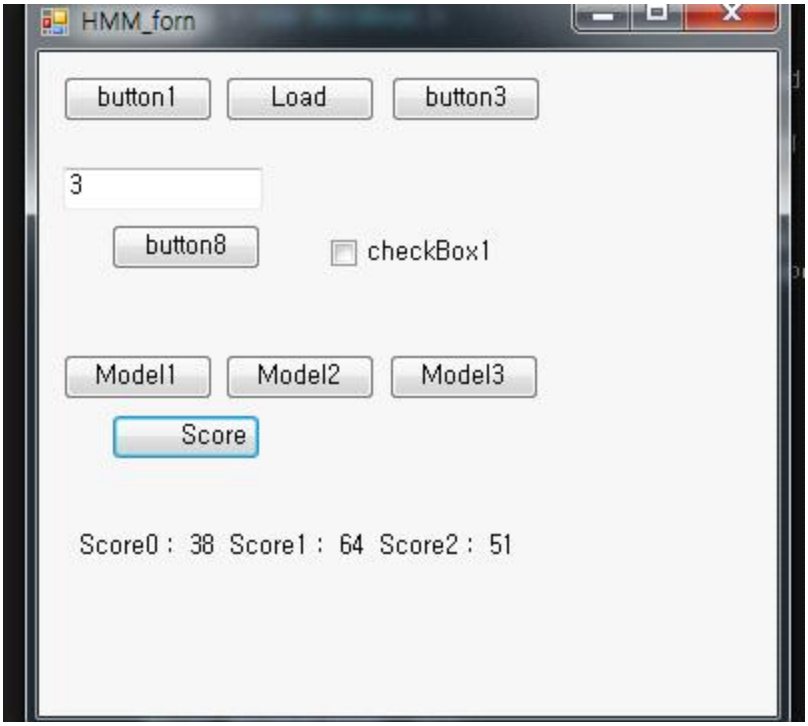
Test에 사용 될 Test Data이며, 랜덤으로 고른 1개의 포켓몬 Go 관련 기사에서 그대로 텍스트만 Copy & Taste하였다.



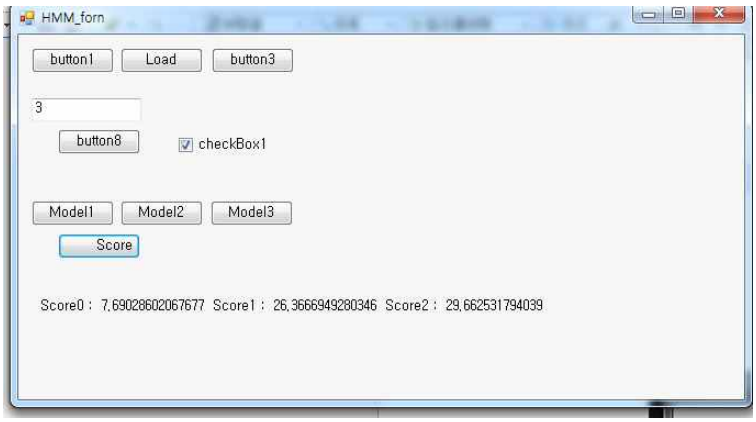
Categorization에 사용 될 Test Data

각 테스트 데이터는 하나의 주제 당 5개 정도의 기사의 Text를 그대로 Copy & Taste하여 구성하였다.

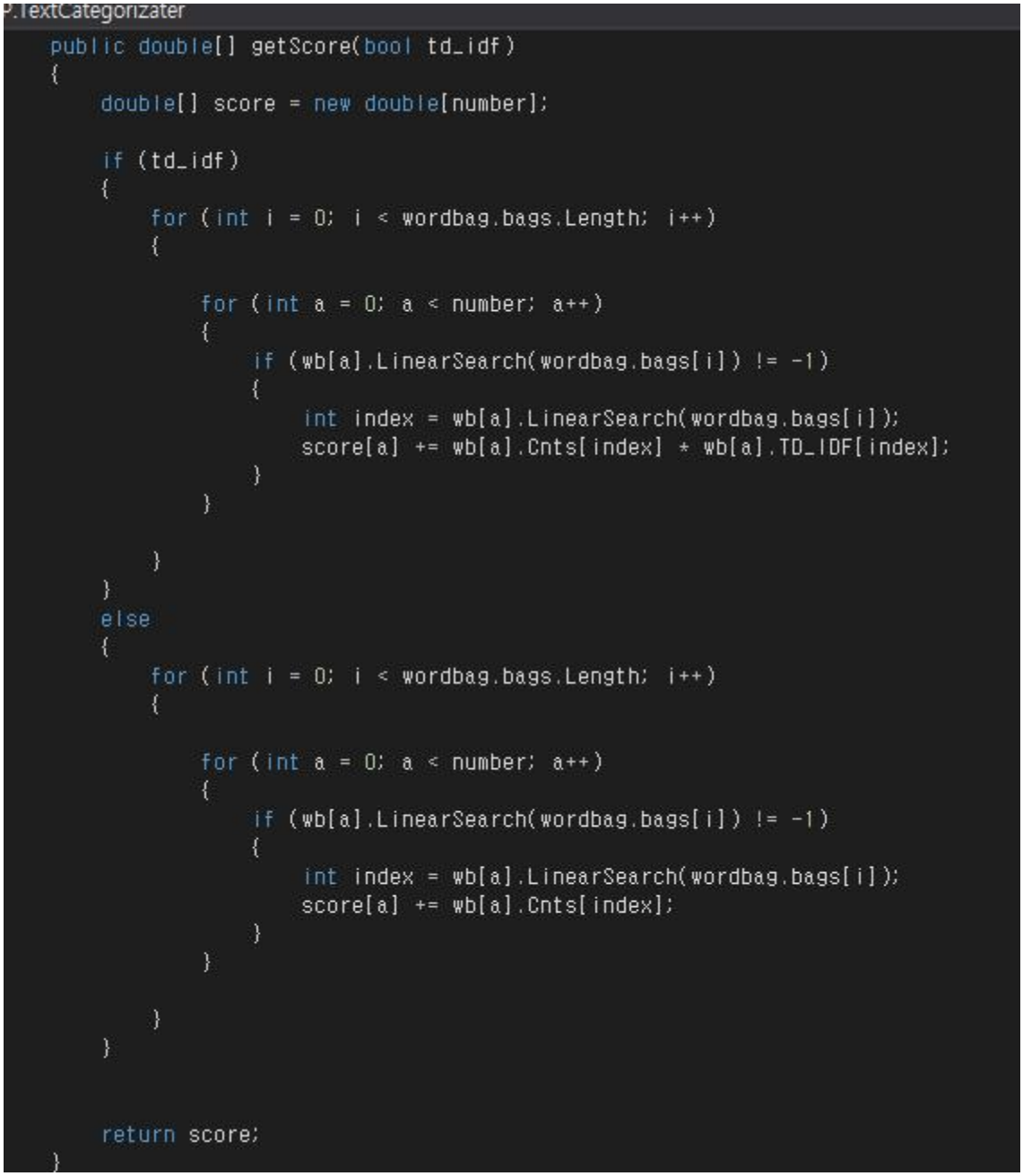
왼쪽 위부터 아이파크 해킹, 정치, 포켓몬고 뉴스이다.



score0부터 아이파크, 해킹, 포켓몬 고에 대한 WordBag을 통해 포켓몬고 뉴스의 Categorization을 위해 조건계산부 확률을 구한 결과이다. 하지만 위 결과에서는 해킹에 관련 된 뉴스로 분류 된 것을 볼 수 있다



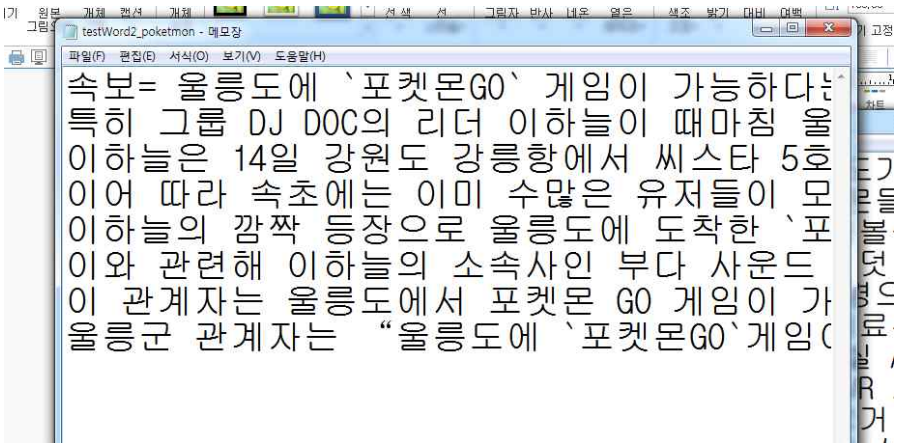
다음 결과는 조건부 계산에 TF-IDF 값을 같이 적용시킨 결과이며, 이전의 잘못 Classification 된 결과에 비해 올바르게 분류함과 동시에 정확도가 크게 향상 된 것을 확인 할 수 있다



```
P.TextCategorizater
    public double[] getScore(bool td_idf)
    {
        double[] score = new double[number];

        if (td_idf)
        {
            for (int i = 0; i < wordbag.bags.Length; i++)
            {

                for (int a = 0; a < number; a++)
                {
                    if (wb[a].LinearSearch(wordbag.bags[i]) != -1)
                    {
                        int index = wb[a].LinearSearch(wordbag.bags[i]);
                        score[a] += wb[a].Cnts[index] * wb[a].TD_IDF[index];
                    }
                }

            }
        }
        else
        {
            for (int i = 0; i < wordbag.bags.Length; i++)
            {

                for (int a = 0; a < number; a++)
                {
                    if (wb[a].LinearSearch(wordbag.bags[i]) != -1)
                    {
                        int index = wb[a].LinearSearch(wordbag.bags[i]);
                        score[a] += wb[a].Cnts[index];
                    }
                }

            }
        }


        return score;
    }
```

TextCategorizer 클래스

TextCategorizer 클래스를 통해 확률을 계산 할 수 있으며, 매개 변수는 tf-idf를 사용할지 안할지에 대한 bool 변수이다



Test에 사용 될 Test Data이며, 랜덤으로 고른 1개의 포켓몬 Go 관련 기사에서 그대로 텍스트만 Copy & Taste하였다.

5. 기대효과 및 활용분야

자연어 처리는 활용도가 매우 높은 최신기술 분야이다. 최근 한국과 네덜란드가 경제적 협력 협약을 맺은 내용이 9시 뉴스에 나온적이 있는데 빅데이터에 관한 내용도 포함되어 있었다. 뉴스에서는 빅데이터의 활용 예시로 SNS의 방대한 텍스트 데이터를 분석해서 마케팅에 사용 될 수 있다는 점을 말했다.

빅데이터 사례 분석을 통한 SNS 마케팅 위기 관리

이와 같은 경우에도 당연히 자연어 처리 기술이 사용되게 되는데 대표적으로 라이브러리에 포함된 기술인 Sentimental Analysis 기술을 통해서 P/N를 알 수 있으면 어떠한 주제나 제품에 대한 반응을 분석 할 수 있는 것이다.

그 외에도 여러 첨단 분야에도 응용되고 있는데 대표적으로 음성인식 기술이다. 음성 인식 기술은 애플의 Siri, 구글 음성인식, 최근 등장한 Nugu 등에서 사용되어지고 있는 음성의 파형을 읽어서 텍스트로 인식하는 기술이다.

Apple Siri

SK Nugu Platform

Google

Google Speech Recognition

위와 같은 경우 사용자가 "야무치게 먹어야지"라고 발음을 했다고 가정하자.
언어의 Context를 나타내주는 Laguage Model에서 만화 케릭터 야무치와 갯벌의 게는 연관성이 크게 없기 때문에 매우 낮은 확률을 나타낼 것이다.

+

하지만 그와 유사한 문장 중에 정준하의 유행어 "야무지게 먹어야지"는 여러 예능 프로그램에서 빈번하게 사용되었기 때문에 문장들을 학습시켰을 때 포함되어 있을 가능성이 높고 그 때문에 Laguage Model에서 비교적 높은 값이 나오게 될 것이다.

그렇게 되면 Speech Recognition 시스템에서는 더 높은 Languagen Model에서의 확률을 가진 문장인 "야무지게 먹어야지"로 보정을 하게 된다.

이외에도 자연어 처리의 응용 분야는 자동 비서 시스템, HRI 등등 응용 분야는 무궁무진하다고 볼 수 있다.

자동 주문 기능

한국의 배달산업은 여러 가지 배달 앱 서비스를 통해 다양하게 발전해가고 있고, 그 트렌드를 생각해서 자동 주문을 적용 시킬 수 있는 클래스를 기능에 추가시켰다.

WordGrammarDictionary 기능을 통해서 여러 가지 자동주문에 관한 함수들을 사용 할 수 있는데 개수와 단위 인식 등의 기능을 사용 할 수 있으며, 제품의 종류와 제품의 옵션을 클래스에 입력하게 되면 주문 텍스트를 읽어들여 주문 결과만 출력해주는 기능도 포함하고 있다. 또한 여러 가지 나올 수 있는 주문들에서 제품의 줄임말 인식 등의 기능 역시 포함하고 있다.

6. 기타(출품작에 대한 추가 설명 및 PT 자료 등 첨부 가능)

아직 미개발 된 기능

Word2Vec

Word2Vec은 구글에서 발표한 1 레이어의 인공신경망을 통해 Language Model을 구현한 것이다. 학습 모델에는 CBOW와 Skip Grams 2가지가 있는데, CBOW는 학습 데이터가 방대 할 때 유리하고, Skip Grams는 적은 데이터에서 유리하다고 한다.
본 프로젝트에서는 CBOW로 Word2Vec을 구현했다.

```
참조 0개
public void LearningContext(int inputVector, int targetVector)
{
    double[,] Der_WeightInput = new double[WordBagLength, NumberOfNodes];
    double[,] Der_WeightOutput = new double[NumberOfNodes, WordBagLength];

    for (int i = 0; i < WordBagLength; i++)
    {
        int t = 0;

        if( i == targetVector ){
            t = 1;
        }

        for (int j = 0; j < NumberOfNodes; j++)
        {
            Der_WeightOutput[j, i] = (result[i] - t) * inputHidden[j];
        }
    }

    for (int j = 0; j < NumberOfNodes; j++)
    {
        double EHi = 0;

        for(int i = 0; i < WordBagLength; i++){
            EHi += result[i] * WeightOutput[j, i];
        }

        Der_WeightInput[inputVector, j] += EHi;
    }

    for (int i = 0; i < WordBagLength; i++)
    {
        for (int j = 0; j < NumberOfNodes; j++)
```

```
참조 0개
public double getProbability(int[] inputVector, int targetVector)
{
    double result = 0;
    results = new double[inputVector.Length, WordBagLength];

    inputHiddens = new double[inputVector.Length, NumberOfNodes];
    outputHiddens = new double[inputVector.Length, WordBagLength];

    for (int k = 0; k < inputVector.Length; k++)
    {
        for (int i = 0; i < NumberOfNodes; i++)
        {
            inputHiddens[k, i] = WeightInput[inputVector[k], i];
        }

        for (int i = 0; i < WordBagLength; i++)
        {
            for (int j = 0; j < NumberOfNodes; j++)
            {
                outputHiddens[k, i] += WeightOutput[j, i] * inputHiddens[k, j];
            }
        }

        double[] temp = Softmax(outputHidden);

        for (int i = 0; i < WordBagLength; i++)
        {
            results[k, i] = temp[i];
        }

    }

    return result;
}
```

Word2Vec Code(Word2Vec 클래스)

기본적인 forward/back propagation C# 코드는 구현이 되어있지만 일부 오류들을 기간 내에 수정하지 못하여, 미완성 기능에 추가시켰다.