# Distance Measurement*

Zhiqiang Li[1], Haoran chen[2], Chencheng Rao[3], and Xupeng Liu[3]

Southeast University

**Abstract.** Distance measurement plays a significant role in the fields of web science and natural language process. It has been used in many applications including words classification, semantics enhancement, search engines. Currently, there are many approaches to measure the distance between two words, such as Hamming distance, edit distance for non-semantic work and some NLP models for semantic work. Some tools created by some advanced universities can also be used to reach our aims. Although there are some limitations for each method, their advantages still play an important role in various fields. This paper, we try to implement four kinds of methods to reach our aim of distance measurement and analyze their features briefly, which can lay the foundation for future study.

**Keywords:** Distance measurement · Semantic · NLP models.

## 1 Problem definition

### 1.1 Problem Description

Given two strings which representing two entities, such as LeBron James and Dwyane Wade, we want to measure their distance automatically and the range of distance is limited from 0 to 1. Smaller the distance is, more related two entities are.

### 1.2 Problem Analyze

The core problem is that what is distance between two words. We tend to divide it into two parts. One is literal distance, which means that the difference between each letter of two compared words, such as format and forbid, which have half of coincidence. In this situation, we do not pay any attentions on the meanings of these two words, just their literally difference. The other is semantic distance. We focus on the real meanings of these two words in this case. For example, LeBron James and Dwyane Wade, which are names of two basketball players. It is meaningless to compare their literal difference. In contrast, other attributes include their background, team, joined competitions and so on, which can represent more information for these two naming entities of NBA players.

* Supported by Peng Wang

## 2 Related works

### 2.1 Algorithms

Hamming distance is used in data transmission error control coding. It can also be used in measure the distance of two words. In comparison, it will punish different letters, word length differences and so on. This is the most basic method for calculating word similarity. In 1965, Russian scientist Vladimir Levenshtein introduced a measurement to indicate the similarity between two words. Editing distance refers to the minimum number of single character editing operations required to convert a word w1 into another word w2 between two words ¡ w1, w2¿. And there are only three single character editing operations defined here. This method can measure the literal distance of words very well. But it cant grasp the inner semantics of words.

### 2.2 Models

In 2013, Tomas Mikolov proposed a predict model named Word2Vec with his team in Google. Word2vec relies on skip-grams or continuous word bag (CBOW) to build neural embedding. Word2vec can express a word into vector form quickly and effectively according to the given corpus through the optimized training model, which provides a new tool for the application research in the field of natural language processing. Stanford NLP experimental group released a paper in 2014, which introduced a new method to generate word vectors. The essence of the model is to integrate the latest global matrix factorization (Matrix Factorization) and local text box capture method (representing Word2Vec). This new method of word vector expression improves the accuracy of many basic tasks of NLP.

### 2.3 Encyclopedia

Wikipedia is a multilingual encyclopedia collaboration project based on Wikipedia technology. It is a multilingual network encyclopedia. Wikipedia has a unique category system in the bottom of each page, which can represent the theme, type or any other relate information for the instance of this page. Many semantic researches use this special function to mine semantic relationships or enhance semantic content.

### 2.4 Tool

WordNet is an English dictionary based on cognitive linguistics designed by psychologists, linguists and computer engineers at Princeton University. It does not just arrange words in alphabetical order, but also forms a ”network of words” according to the meaning of words. This network is mainly composed of hypernym and hyponym relations and synonym relations. Many knowledge system (YAGO, DBPedia) use it to contribute to their constructions.

## 3 Approach overview

### 3.1 Non-semantic measurements

Inspired by related work, we select and implement Hamming distance and Edit distance by using Python to compare literal distance. We also try to use some representative samples to test the perform of these two algorithms.

### 3.2 Semantic measurements

As for semantic distance, our idea is converting words into vectors with semantics. So, we choose Word2Vec and GloVe model to assist us in accomplishing our aims. After getting vectors, we can calculate vector cosine distance easily. There is a famous distance called Jaccard distance, which fully use union and intersection composed of the attributes of entities to measure the distance. The key is how to find the attributes set of two compared entities. We purpose to use category system from Wikipedia pages and extract the categories to form the attribute sets. Another method is to use complete tool to measure the distance. We select a widely used tool WordNet to achieve our goals.

## 4 Theories of measurements

### 4.1 Hamming distance

Given two strings (denoted as s and t), Hamming distance is used to directly calculate the corresponding positions of characters in two strings. Besides, the difference of length will also be taken into consideration. Basing on this thought, we have below formula:
$HammingDist(s,t) = \frac{(\sum_{i=1}^{min(|s|,|t|)} s[i] \neq t[i]) + ||s|-|t||}{max(|s|,|t|)}$
For instance, two strings harmony (devoted as s) and harmonious (devoted as t). We get maximal length is 10. Because it is longer than s, we get . Then we compare each letter between s and t. Only at the last character of s, we find the letter y which is different from t in the correspond position with a letter i, we punish this situation. Finally, we calculate the Hamming distance in accords with formula and get the result which is 6/10.

### 4.2 Edit distance

Given two strings (denoted as a and b), Edit distance is used to convert a to b by using three modes. Three modes are insertion, deletion and substitution. Such as boy and toys, we need to substitute b into t and then add s behind y. The min edit distance in this example is 2. The formula is below: We iterate through two strings from their last letter. Parameters i and j represent current iteration index. Originally, i equals the length of a and j equals the length of b. The algorithm compares a i and b j. In this situation, we have three methods to

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

process, one is iterating string a forward (i minus one) and rise the distance by one (mode Insertion), then the problem become smaller which we calculate new modified string a and string b. In another method, we can also iterate string b forward (j minus one) and rise the distance by one (mode Deletion), then the problem become smaller which we calculate string a and new modified string b. Last, we can use mode substitution to substitute a i and b j directly. If a i equals with b j, distance does not need to add one, otherwise distance need to add one. We can get both modified strings which are shorter than before. After processing this step, we detect whether i or j equals zero, if so, we assign current distance as i or j which has bigger value. This is the core idea of this algorithm, we cite an example to describe the procedure of dynamic programming. We have two words may and rain. This is iteration tree:

Each leaf means one method to finish measuring edit distance. We count all weights from leaf node to root node and get the edit distance when using this method. We can get minimal edit distance from the tree. We also find that the ways to get minimal distance are not only one in some cases.

### 4.3 Word2Vec

The model can be realized in two train methods including skip-gram and CBOW. We introduce skip-gram here. Skip-gram predict the surrounding words according to the center word W(t). As shown in Figure 2, the structure of Skip-gram is a neural network model with one layer. The model use one-hot representation vector form of word (devoted as w input) as input and the dimension is V1, V represents the whole quantities of words in corpus. Between input layer and hidden layer, a matrix (devoted as W) whose dimension is dV, which is introduced. Parameter d is man-specific dimension for dimension reduction. Matrix W will be trained and used to right multiply with w input. After calculating, we will get a new vector V c whose dimension is d1. V c is semantic vector, which can be used to calculate distance through cosine calculation. The process between hidden layer and output layer aims to predict surrounding words. We introduce a new matrix (Vd) to be trained and left multiply with V c . We get the output V output (V1) with form of one-hot representation vector. Due to some values in matrix are negative, we use softmax function to process normalization and get vectors V new-output. Finally, we use predicted vectors V new-output and actual vectors V actual to acquire errors and process back propagation to train

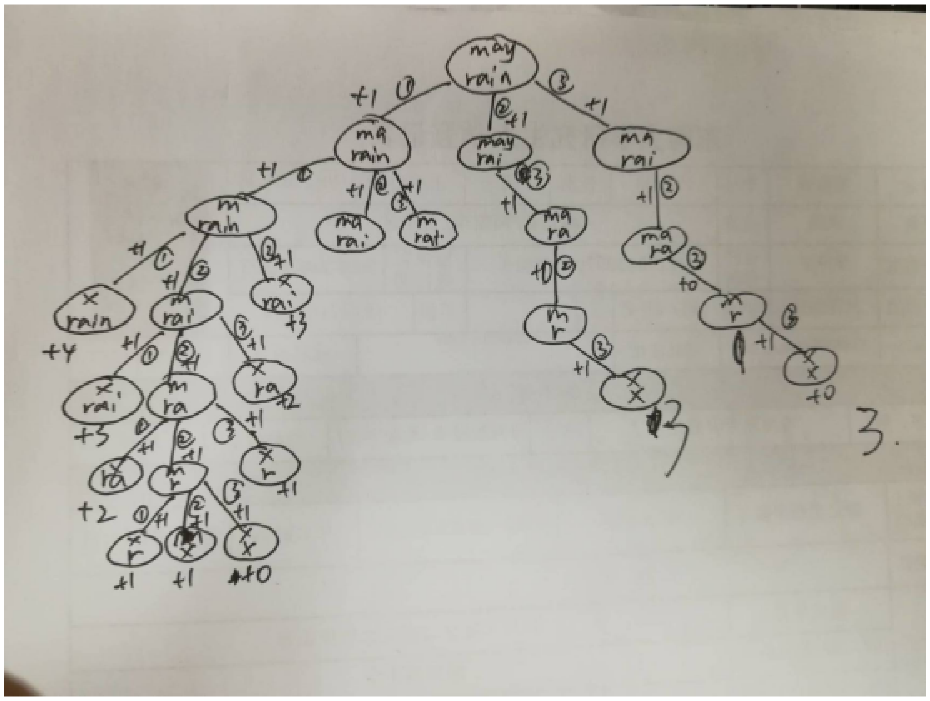


**Fig. 1.** Iteration tree when process may and rain.

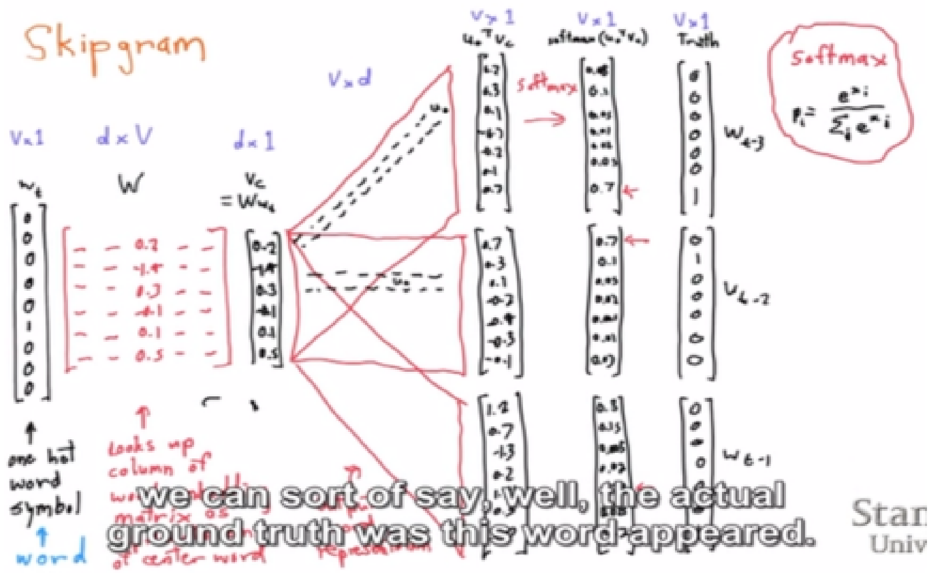

Skipgram
Softmax
we can sort of say, well, the actual ground truth was this word appeared.

**Fig. 2.** The structure of Skip-gram

this model. All in all, what we want is an intermediate product in the process of predicting. The model provides vector with semantics for us.

### 4.4 GloVe

GloVe is a little modification of Word2Vec. They are fundamentally similar. Both of them capture local co-occurrence statistics(neighbors) and capture distance between embedding vector(analogies). The biggest difference of GloVe and Word2Vec is that GloVe is a count-based model while Word2Vec is a predictive model. As mentioned before, there mainly exists two kinds of predictive model: predict middle vocabulary according to the context (CBOW); the context is predicted according to the middle vocabulary (Skip-gram). On the other hand,

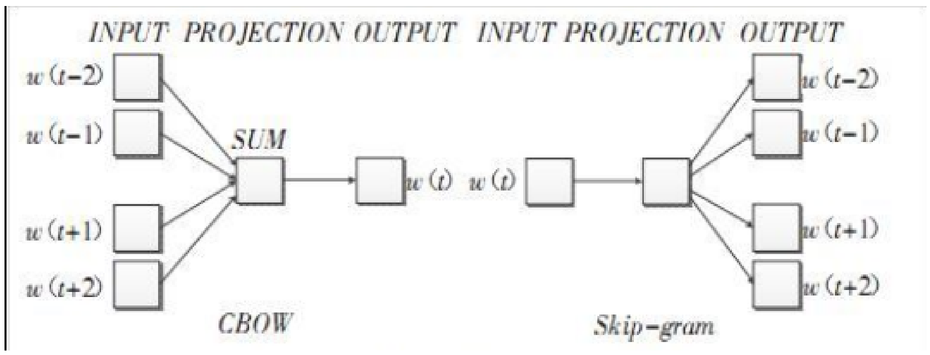


**Fig. 3.** Two main predictive model: CBOW and Skip-gram.

count-based model constructs a co-occurrence matrix to count the relevance of words. But co-occurrence matrix is large and sparse. It is necessary to reduce the dimension of the matrix. GloVe is typically a count-based model. Firstly, a co-occurrence matrix is constructed from the corpus, and each element in the matrix represents the number of times the word and the context word co-occur in a window with a specific size. In general, the minimum unit of this number is 1. But in Glove, it proposes a decreasing weighting function: for calculating the weight based on the distance of the two words in the context window. That is to say, the farther the distance is the two words, the smaller the weight of the total count. Secondly, it constructs the approximate relationship between the word vector and the co-occurrence matrix. The author proposes that the following formula can approximate the relationship between the word vector and the co-occurrence matrix:
$w_i^T \tilde{w_j} + b_i + \tilde{b_j} = log(X_{i,j})$
and is the word vector we want; and is the bias term of the word vector respectively. The last step is to construct the loss function:
$J = \sum_{i,j=1}^{V} f(X_{i,j})(w_i^T \tilde{w_j} + b_i + \tilde{b_j} - log(X_{i,j}))^2$

The basic format of loss function is mean square loss. In addition, a weighting function is added to it, which requires: (1) The weight of these words is greater than rare co-occurrences, so this function is a non-decreasing function; (2) We dont want this weight to be too big. When it reaches a certain level, it should not increase; (3) If two words do not appear together, that is, , then they should not participate in the calculation of the Loss function, that is, should satisfy . So, the function can be constructed as:

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha} & if\, x < x_{max} \\ 1 & otherwise. \end{cases}$$

In the formula, alpha is decay factor.

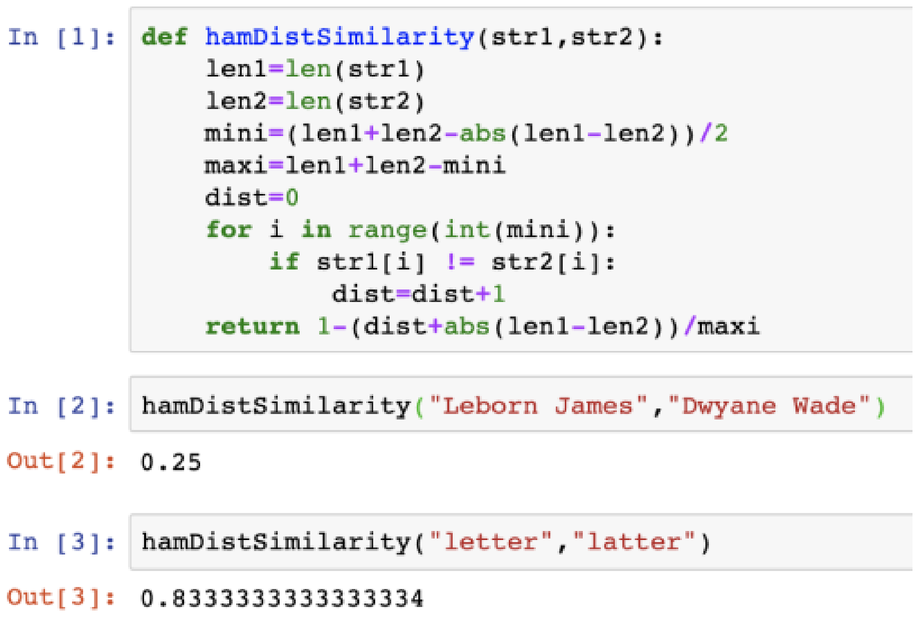
## 5 Experiments

### 5.1 Algorithms Implement

Firstly, we realize the program of Hamming distance. From the result of first group we can find that the distance of Leborn James and Dwyane Wade is 0.25. Low distance means low relevant literally between these two words. In contrast, the result of comparing letter and latter is high, just as we can see that only one letter is different. Through comparison between two groups, we can find that this method just takes advantages in literal difference recognition and evaluation. However, insertion and deletion are not taken into consideration in this method. In the first group of samples, we know that James and Wade are all basketball players and they were teammates before, but the relevance is low, which means that semantics are also not taken seriously.

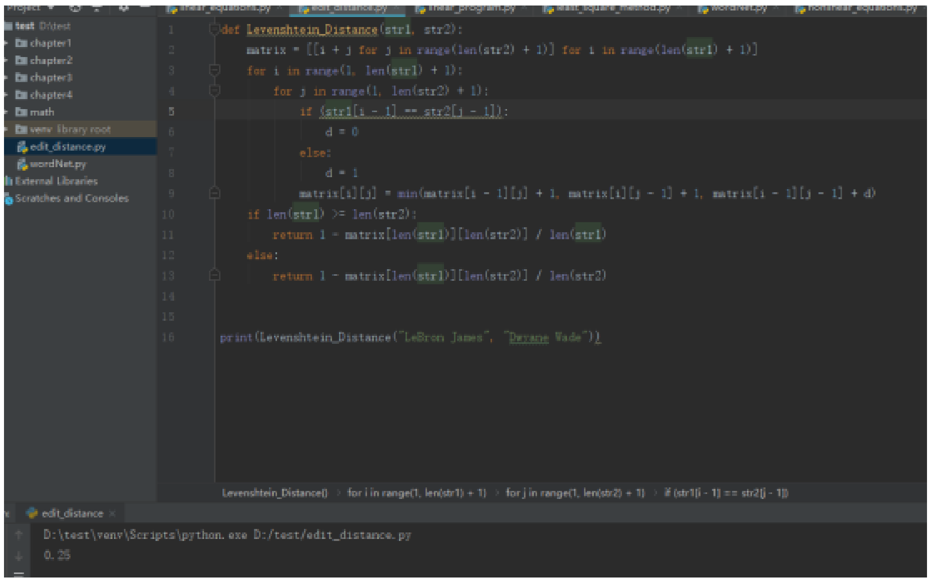
Then, we try to measure Edit distance. The code is shown in Figure 5:

We also cite LeBron James and Dwayne Wade as examples. We get the minimal edit distance is 9 and maximal edit distance is 12. So, we get the result which is 0.25, which indicate that the relevance between two examples is low. This algorithm introduces three modes for converting and uses dynamic programming to find the best approach to convert one word to the other, which other algorithms cant reach in literal layer. But the drawback is obviously that it just stays on the literal level like Hamming distance, no semantic information is revealed.



```
In [1]: def hamDistSimilarity(str1,str2):
            len1=len(str1)
            len2=len(str2)
            mini=(len1+len2-abs(len1-len2))/2
            maxi=len1+len2-mini
            dist=0
            for i in range(int(mini)):
                if str1[i] != str2[i]:
                    dist=dist+1
            return 1-(dist+abs(len1-len2))/maxi

In [2]: hamDistSimilarity("Leborn James","Dwyane Wade")

Out[2]: 0.25

In [3]: hamDistSimilarity("letter","latter")

Out[3]: 0.8333333333333334
```

**Fig. 4.** The results of Hamming Distance measurement.



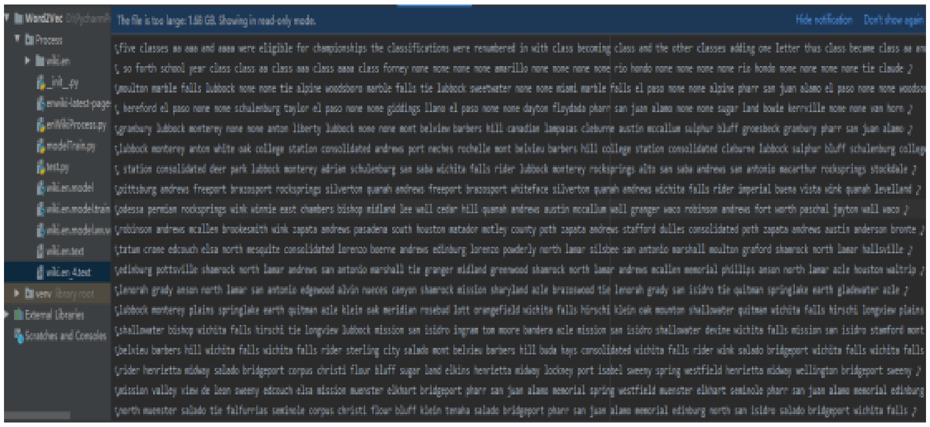
```
def Levenshtein_Distance(str1, str2):
    matrix = [[i + j for j in range(len(str2) + 1)] for i in range(len(str1) + 1)]
    for i in range(1, len(str1) + 1):
        for j in range(1, len(str2) + 1):
            if (str1[i - 1] == str2[j - 1]):
                d = 0
            else:
                d = 1
            matrix[i][j] = min(matrix[i - 1][j] + 1, matrix[i][j - 1] + 1, matrix[i - 1][j - 1] + d)
    if len(str1) >= len(str2):
        return 1 - matrix[len(str1)][len(str2)] / len(str1)
    else:
        return 1 - matrix[len(str1)][len(str2)] / len(str2)


print(Levenshtein_Distance("LeBron James", "Dwyane Wade"))
```

```
D:\test\venv\Scripts\python.exe D:/test/edit_distance.py
0.25
```

**Fig. 5.** The experiment of Edit distance.

### 5.2 Models Training

We implement two models in this section. We train Word2Vec model at first. The procedure is divided into three parts including corpus downloading and preprocessing, package installation and training, sample running. We choose Wikipedia English Corpus as our train materials. Because the corpus is too big (15.4G) to finish model training in a limited time, we divide corpus into eight parts and select one to be trained, as shown in Figure 6.

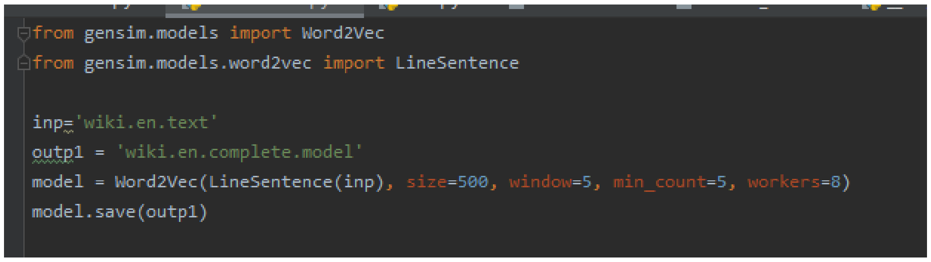


The file is too large: 1.68 GB. Showing in read-only mode.

```
five classes aa aaa and aaaa were eligible for championships the classifications were renumbered in with class becoming class and the other classes adding one letter thus class became class aa and
so forth school year class class aa class aaa class aaaa class forney none none none none amarillo none none none none rio hondo none none none none rio hondo none none none none tie claude
moulton marble falls lubbock none none tie alpine woodsboro marble falls tie lubbock sweetwater none none miami marble falls el paso none none alpine pharr san juan alamo el paso none none woodsbo
hereford el paso none none schulenburg taylor el paso none none giddings llano el paso none none dayton floydada pharr san juan alamo none none sugar land bowie kerrville none none van horn
granbury lubbock monterey none none anton liberty lubbock none none mont belvieu barbers hill canadian lampasas cleburne austin mccallum sulphur bluff groesbeck granbury pharr san juan alamo
lubbock monterey anton white oak college station consolidated andrews port neches rochelle mont belvieu barbers hill college station consolidated cleburne lubbock sulphur bluff schulenburg college
station consolidated deer park lubbock monterey adrian schulenburg san saba wichita falls rider lubbock monterey rocksprings alto san saba andrews san antonio macarthur rocksprings stockdale
pittsburg andrews freeport brazosport rocksprings silverton quanah andrews freeport brazosport whiteface silverton quanah andrews wichita falls rider imperial buena vista wink quanah levelland
odessa permian rocksprings wink winnie east chambers bishop midland lee wall cedar hill quanah andrews austin mccallum wall granger waco robinson andrews fort worth paschal jayton wall waco
robinson andrews mcallen brookesmith wink zapata andrews pasadena south houston matador motley county poth zapata andrews stafford dulles consolidated poth zapata andrews austin anderson bronte
tatum crane edcouch elsa north mesquite consolidated lorenzo boerne andrews edinburg lorenzo powderly north lamar silsbee san antonio marshall moulton graford shamrock north lamar hallsville
edinburg pottsville shamrock north lamar andrews san antonio marshall tie granger midland greenwood shamrock north lamar andrews mcallen memorial phillips anson north lamar azle houston waltrip
lenorah grady anson north lamar san antonio edgewood alvin nueces canyon shamrock mission sharyland azle brazoswood tie lenorah grady san isidro tie quitman springlake earth gladewater azle
lubbock monterey plains springlake earth quitman azle klein oak meridian rosebud lott orangefield wichita falls hirschi klein oak mounton shallowater quitman wichita falls hirschi longview plains
shallowater bishop wichita falls hirschi tie longview lubbock mission san isidro ingram tom moore bandera azle mission san isidro shallowater devine wichita falls mission san isidro stamford mont
belvieu barbers hill wichita falls wichita falls rider sterling city salado mont belvieu barbers hill buda hays consolidated wichita falls rider wink salado bridgeport wichita falls wichita falls
rider henrietta midway salado bridgeport corpus christi flour bluff sugar land elkins henrietta midway lockney port isabel sweeny spring westfield henrietta midway wellington bridgeport sweeny
mission valley view de leon sweeny edcouch elsa mission muenster elkhart bridgeport pharr san juan alamo memorial spring westfield muenster elkhart seminole pharr san juan alamo memorial edinburg
north muenster salado tie falfurrias seminole corpus christi flour bluff klein tenaha salado bridgeport pharr san juan alamo memorial edinburg north san isidro salado bridgeport wichita falls
```

**Fig. 6.** The part of content from corpus.

Because Word2vec model has been integrated by gensim package, we install gensim and apply internal methods to train the model.

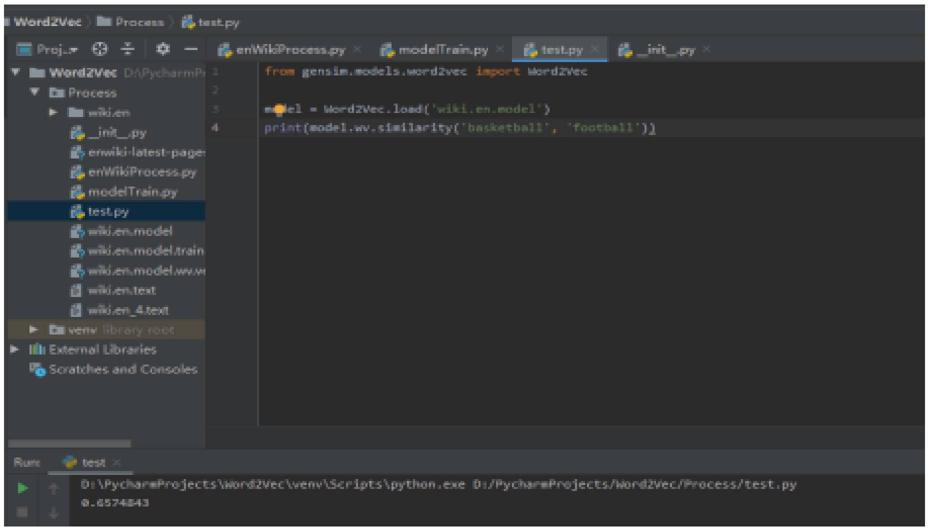


```python
from gensim.models import Word2Vec
from gensim.models.word2vec import LineSentence

inp='wiki.en.text'
outp1 = 'wiki.en.complete.model'
model = Word2Vec(LineSentence(inp), size=500, window=5, min_count=5, workers=8)
model.save(outp1)
```

**Fig. 7.** The code of training Word2Vec.

After acquiring trained model, we select three sports entities football, swimming and basketball to test.

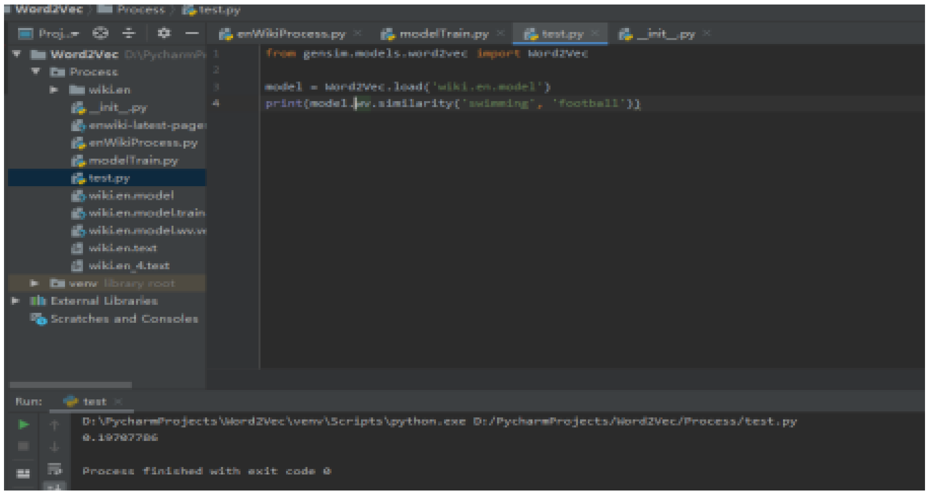
We can get that the distance between football and basketball is larger than football and swimming, which can deduce that the former is more relevant. There



```python
from gensim.models.word2vec import Word2Vec

model = Word2Vec.load('wiki.en.model')
print(model.wv.similarity('basketball', 'football'))
```

```
D:\PycharmProjects\Word2Vec\venv\Scripts\python.exe D:/PycharmProjects/Word2Vec/Process/test.py
0.6574843
```

**Fig. 8.** Similarity calculation by Word2Vec(1).

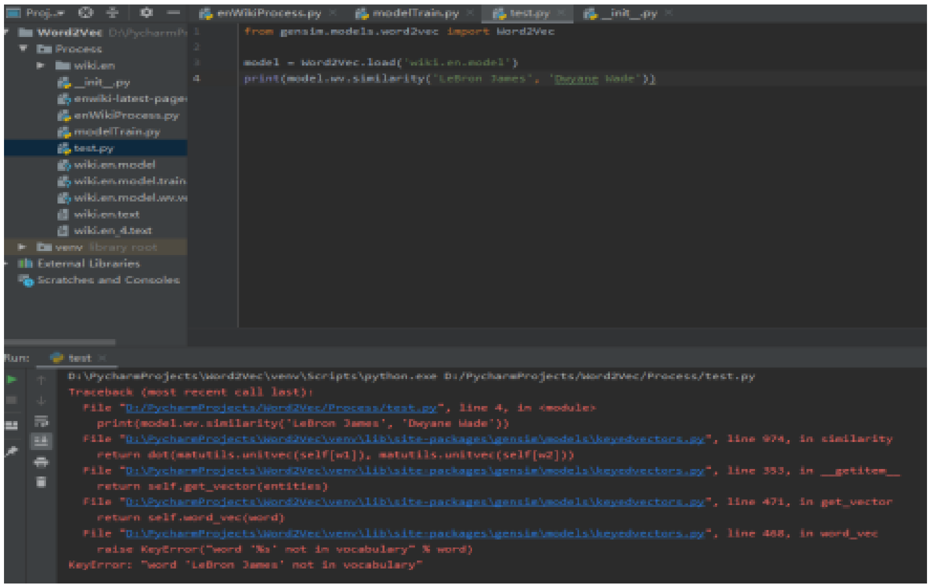


```python
from gensim.models.word2vec import Word2Vec

model = Word2Vec.load('wiki.en.model')
print(model.wv.similarity('swimming', 'football'))
```

```
D:\PycharmProjects\Word2Vec\venv\Scripts\python.exe D:/PycharmProjects/Word2Vec/Process/test.py
0.19707786

Process finished with exit code 0
```

**Fig. 9.** Similarity calculation by Word2Vec(2).

also existing a problem that it cant recognize entity comprised more than two words. Moreover, name entity cant be recognized.



```
from gensim.models.word2vec import Word2Vec

model = Word2Vec.load('wiki.en.model')
print(model.wv.similarity('LeBron James', 'Dwyane Wade'))
```

```
D:\PycharmProjects\Word2Vec\venv\Scripts\python.exe D:/PycharmProjects/Word2Vec/Process/test.py
Traceback (most recent call last):
  File "D:/PycharmProjects/Word2Vec/Process/test.py", line 4, in <module>
    print(model.wv.similarity('LeBron James', 'Dwyane Wade'))
  File "D:\PycharmProjects\Word2Vec\venv\lib\site-packages\gensim\models\keyedvectors.py", line 974, in similarity
    return dot(matutils.unitvec(self[w1]), matutils.unitvec(self[w2]))
  File "D:\PycharmProjects\Word2Vec\venv\lib\site-packages\gensim\models\keyedvectors.py", line 353, in __getitem__
    return self.get_vector(entities)
  File "D:\PycharmProjects\Word2Vec\venv\lib\site-packages\gensim\models\keyedvectors.py", line 471, in get_vector
    return self.word_vec(word)
  File "D:\PycharmProjects\Word2Vec\venv\lib\site-packages\gensim\models\keyedvectors.py", line 468, in word_vec
    raise KeyError("word '%s' not in vocabulary" % word)
KeyError: "word 'LeBron James' not in vocabulary"
```

**Fig. 10.** Error match with name entities.

Another model that we implement is GloVe. We use the package of tf-glove, which git from Github. The dataset we use comes from Amazon Food Comments [Kaggle], which contains 568,454 food comments left by Amazon users before October 2012. The model runs under the framework of TensorFlow. We use the comments column to train the mode. Then follow the guidance provided by the project, we successfully train the model and check the results.

We also use TensorBoard to illustrate the graph of model and the training process. From the statistic, we can conclude that the model converges at approximately 2.9M steps. After training the model, we get the vector of each word. To display the results, we compare the cosine similarity of couple of words and show the words in 2D vector space.

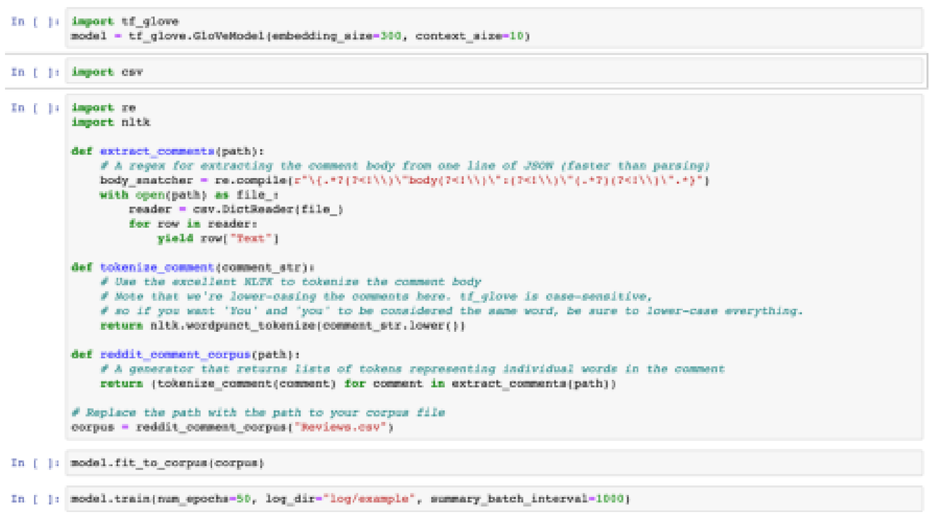
## 5.3 Tool Search

We use WordNet to process experiment. Firstly, we import nltk package and use it to download WordNet data source. Then, it is convenient for us to calculate and compare the distance. We choose three words basketball, soccer and football. The results are shown in Figure 15 and 16.

We also can get the results that soccer is more relevant with football than basketball.

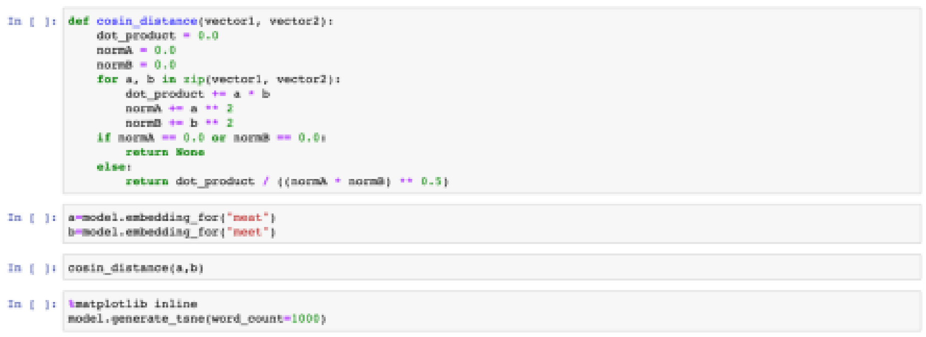
[illegible]

**Fig. 11.** Amazon Food Comments Dataset.



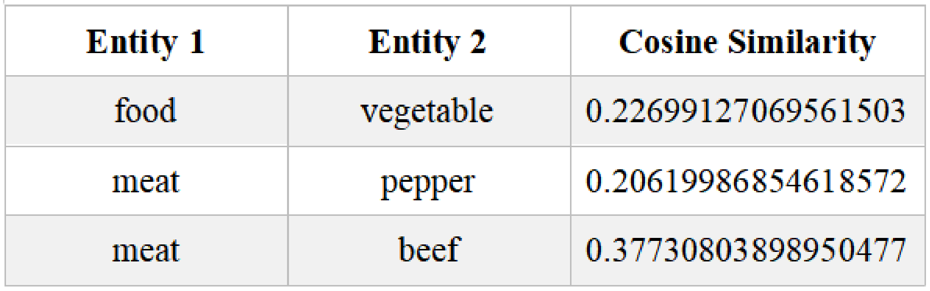
```
In [ ]: import tf_glove
        model = tf_glove.GloVeModel(embedding_size=300, context_size=10)

In [ ]: import csv

In [ ]: import re
        import nltk

        def extract_comments(path):
            # A regex for extracting the comment body from one line of JSON (faster than parsing)
            body_snatcher = re.compile(r"\{.*?(?<!\\)\"body(?<!\\)\":(?<!\\)\"(.*?)(?<!\\)\".*}")
            with open(path) as file_:
                reader = csv.DictReader(file_)
                for row in reader:
                    yield row["Text"]

        def tokenize_comment(comment_str):
            # Use the excellent NLTK to tokenize the comment body
            # Note that we're lower-casing the comments here. tf_glove is case-sensitive,
            # so if you want 'You' and 'you' to be considered the same word, be sure to lower-case everything.
            return nltk.wordpunct_tokenize(comment_str.lower())

        def reddit_comment_corpus(path):
            # A generator that returns lists of tokens representing individual words in the comment
            return (tokenize_comment(comment) for comment in extract_comments(path))

        # Replace the path with the path to your corpus file
        corpus = reddit_comment_corpus("Reviews.csv")

In [ ]: model.fit_to_corpus(corpus)

In [ ]: model.train(num_epochs=50, log_dir="log/example", summary_batch_interval=1000)
```

**Fig. 12.** Code of the preproduce of dataset, training, showing results(1).



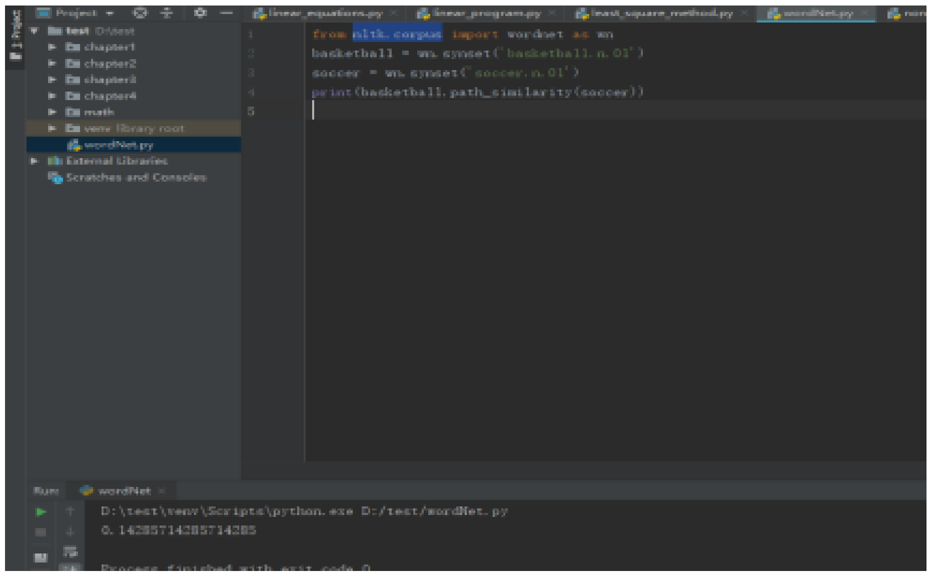
```
In [ ]: def cosin_distance(vector1, vector2):
            dot_product = 0.0
            normA = 0.0
            normB = 0.0
            for a, b in zip(vector1, vector2):
                dot_product += a * b
                normA += a ** 2
                normB += b ** 2
            if normA == 0.0 or normB == 0.0:
                return None
            else:
                return dot_product / ((normA * normB) ** 0.5)

In [ ]: a=model.embedding_for("meat")
        b=model.embedding_for("meet")

In [ ]: cosin_distance(a,b)

In [ ]: %matplotlib inline
        model.generate_tsne(word_count=1000)
```

**Fig. 13.** Code of the preproduce of dataset, training, showing results(2).



| Entity 1 | Entity 2 | Cosine Similarity |
|---|---|---|
| food | vegetable | 0.22699127069561503 |
| meat | pepper | 0.20619986854618572 |
| meat | beef | 0.37730803898950477 |

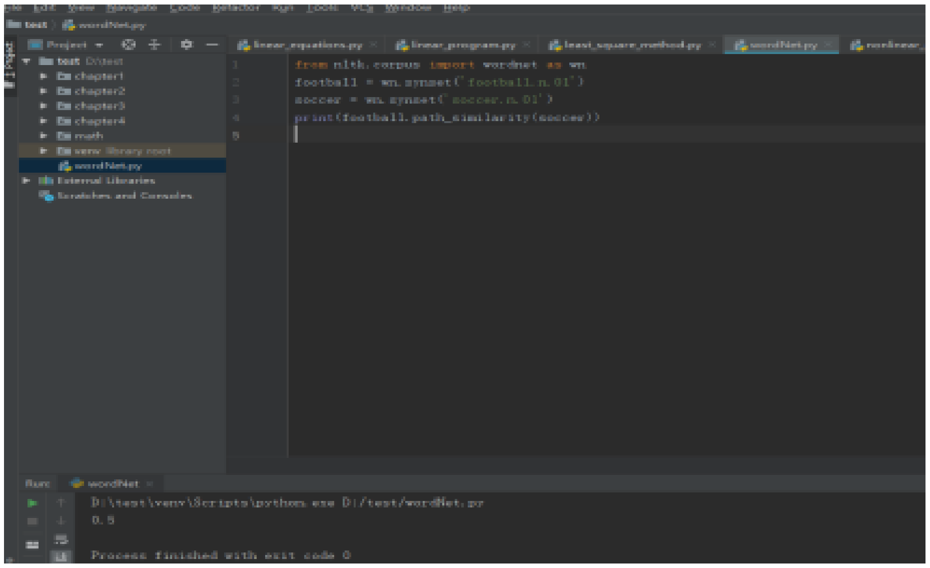
**Fig. 14.** Cosine Similarity of words.



```
from nltk.corpus import wordnet as wn
basketball = wn.synset('basketball.n.01')
soccer = wn.synset('soccer.n.01')
print(basketball.path_similarity(soccer))
```

```
D:\test\venv\Scripts\python.exe D:/test/wordNet.py
0.14285714285714285

Process finished with exit code 0
```

**Fig. 15.** Comparison between two words by WordNet(1).



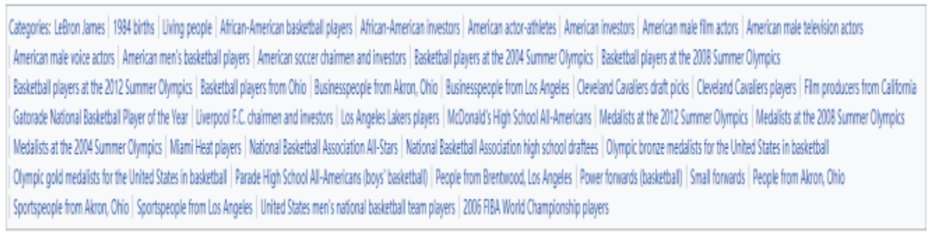
```
from nltk.corpus import wordnet as wn
football = wn.synset('football.n.01')
soccer = wn.synset('soccer.n.01')
print(football.path_similarity(soccer))
```

```
D:\test\venv\Scripts\python.exe D:/test/wordNet.py
0.5

Process finished with exit code 0
```

**Fig. 16.** Comparison between two words by WordNet(2).

## 5.4 Wikipedia Extraction

We use category system in Wikipedia to compose the attribute sets (denoted as A and B) of two entities, which support the calculation of Jaccard distance. The formula is $J(A,B) = \frac{|A \bigcap B|}{|A \cup B|}$. We search Wikipedia for two entities LeBron James and Dwyane Wade and get the categories shown in Figure 17 and 18.



Categories: LeBron James | 1984 births | Living people | African-American basketball players | African-American investors | American actor-athletes | American investors | American male film actors | American male television actors
| American male voice actors | American men's basketball players | American soccer chairmen and investors | Basketball players at the 2004 Summer Olympics | Basketball players at the 2008 Summer Olympics
| Basketball players at the 2012 Summer Olympics | Basketball players from Ohio | Businesspeople from Akron, Ohio | Businesspeople from Los Angeles | Cleveland Cavaliers draft picks | Cleveland Cavaliers players | Film producers from California
| Gatorade National Basketball Player of the Year | Liverpool F.C. chairmen and investors | Los Angeles Lakers players | McDonald's High School All-Americans | Medalists at the 2012 Summer Olympics | Medalists at the 2008 Summer Olympics
| Medalists at the 2004 Summer Olympics | Miami Heat players | National Basketball Association All-Stars | National Basketball Association high school draftees | Olympic bronze medalists for the United States in basketball
| Olympic gold medalists for the United States in basketball | Parade High School All-Americans (boys' basketball) | People from Brentwood, Los Angeles | Power forwards (basketball) | Small forwards | People from Akron, Ohio
| Sportspeople from Akron, Ohio | Sportspeople from Los Angeles | United States men's national basketball team players | 2006 FIBA World Championship players

**Fig. 17.** Categories of LeBron James from Wikipedia.

After extracting these categories as their own attribute set, we can use the formula of Jaccard distance to measure. Because of time limited, we have not finished the program of this method. This method also has some drawbacks. One is that the categories are made by people, which means that the precision of these information may not be so high.

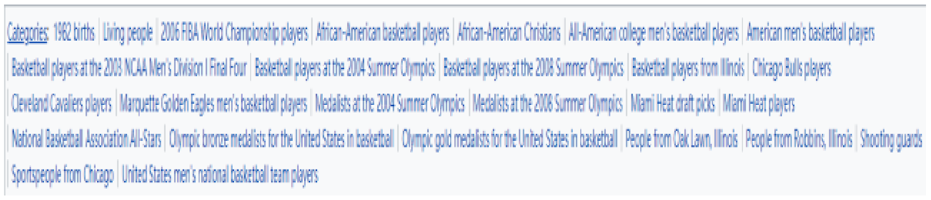

Categories: 1982 births | Living people | 2006 FIBA World Championship players | African-American basketball players | African-American Christians | All-American college men's basketball players | American men's basketball players | Basketball players at the 2003 NCAA Men's Division I Final Four | Basketball players at the 2004 Summer Olympics | Basketball players at the 2008 Summer Olympics | Basketball players from Illinois | Chicago Bulls players | Cleveland Cavaliers players | Marquette Golden Eagles men's basketball players | Medalists at the 2004 Summer Olympics | Medalists at the 2008 Summer Olympics | Miami Heat draft picks | Miami Heat players | National Basketball Association All-Stars | Olympic bronze medalists for the United States in basketball | Olympic gold medalists for the United States in basketball | People from Oak Lawn, Illinois | People from Robbins, Illinois | Shooting guards | Sportspeople from Chicago | United States men's national basketball team players

**Fig. 18.** Categories of Dwyane Wade from Wikipedia.

## 6 Conclusion

In this paper, we use four methods to solve the problem of measuring distance. Hamming distance and Edit distance are effective to measure literal distance, but they have no effect on semantics. Word2Vec and GloVe try to use corpus to train model and acquire representation of words by vectors with semantics. They fuse semantic information, but they cant solve the situation which entity is composed of more than one word. WordNet is a useful tool to find semantic distance and it is very convenient. In the last, we try to extract categories of each entity from Wikipedia category system and use them to calculate Jaccard distance. The realization of this method will be complete in the future work. Through this task, our understanding of distance is more profound. Code implementation and model training of the algorithm are more skilled. Our team has gained a lot.

## References

1. Hjelmqvist, Sten (26 March 2012), Fast, memory efficient Levenshtein algorithm
2. Guthrie, D., Allison, B., Liu, W., Guthrie, L., Wilks, Y.(2006, May): A closer look at skip-gram modelling. In LREC (pp. 1222-1225)
3. Pennington, J., Socher, R., Manning, C. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).
4. Baroni, M., Dinu, G., Kruszewski, G. (2014, June). Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 238-247).
5. Wu, T., Qi, G., Luo, B., Zhang, L., Wang, H. (2019). Language-Independent Type Inference of the Instances from Multilingual Wikipedia. International Journal on Semantic Web and Information Systems (IJSWIS), 15(2), 22-46.