**Techniques for word/document embedding**

1. **Bag-of-words**



“dog eat dog world, baby!” will be represented by a 550-length vector *v* (assuming a vocabulary of 550 words was chosen), V₇₆=1, as the 76th word of the vocabulary is *world*.

* V₂₀₀=2, as the 200th word of the vocabulary is *dog*.
* V₃₂₂=1, as the 332nd word of the vocabulary is eat.
* The word *baby* was not selected for the inclusion in the vocabulary, so it induces a value of 1 on no entry of the vector.

1. **Document-Term Matrix?**

In the context of NLP, a text corpus is simply a collection of documents and a document is a collection of a sentence, paragraphs or words.

D1: NLP is super cool to learn and apply.

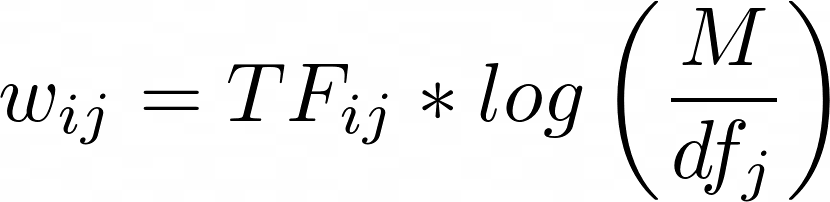
D2: I am currently learning NLP, you can too!

D3: CS224n at Stanford is the best NLP class you can ever take!

 A term that occurs *frequently* in the text corpus will have a much *higher*score, and a term that occurs *rarely* in the text corpus will have a much *lower* score.

In populating the Document-Term Matrix with the number of occurrences, frequently occurring terms are assigned a higher score than the rarely occurring terms.

TF-IDF score is a combination of two metrics: the **Term Frequency (TF)** and the **Inverse Document Frequency (IDF)**.



* TF\_ij is the number of times the term Tj occurs in the document  Di
* df\_j is the number of documents containing the term TF\_ij
* M is the total number of documents,

**Python code for generating document term matrix in Python**

from sklearn.feature\_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop\_words='english',smooth\_idf=True)

input\_matrix = vectorizer.fit\_transform(text)

print(input\_matrix)

input\_matrix = vectorizer.fit\_transform(text).todense()