16.1 What is NLP

* We will be looking at two such representations.
  1. Sparse representation with **term-term co-occurrence matrix:** The idea is to go through a corpus of text, keeping a count of all of the words that appear in context of each word (within a window
* Term-term co-occurrence matrices are long and sparse.
  + length |V| is usually large (e.g., > 50,000)
  + most elements are zero
* OK because there are efficient ways to deal with sparse matrices.

Alternative

* Learn short (~100 to 1000 dimensions) and dense vectors.
* Short vectors may be easier to train with ML models (less weights to train).
* They may generalize better.
* In practice they work much better!

**Background pattern

Description automatically generated**

* 1. Dense representation with **Word2Vec:** A family of algorithms to create dense word embedding:
     + Able to capture complex relationships between words.

Text

Description automatically generated

* Both are based on two ideas: **distributional hypothesis** and **vector space model**.

### Distributional hypothesis

You shall know a word by the company it keeps.

Firth, 1957

If A and B have almost identical environments we say that they are synonyms.Harris, 1954

Example:

* Her **child** loves to play in the playground.
* Her **kid** loves to play in the playground.

### Vector space model

* Model the meaning of a word by placing it into a vector space.
* A standard way to represent meaning in NLP
* The idea is to create **embeddings of words** so that distances among words in the vector space indicate the relationship between them.

Topics Modelling

### Topic modeling: Input and output

* Input
  + A large collection of documents
  + A value for the hyperparameter K (e.g., K=3)
* Output
  + For each topic, what words describe that topic?
  + For each document, what topics are expressed by the document?

Diagram

Description automatically generated

16.2 Text preprocessing

### Tokenization

* Sentence segmentation
  + Split text into sentences
* Word tokenization
  + Split sentences into words

Types and tokens

~ Type: an element in the vocabulary

~ Token: an instance of that type in running text

Lemmatization: want to ignore some morphological differences between works. E.g if your search term is "studying for ML quiz" you might want to include pages containing "tips to study for an ML quiz" or "here is how I studies for my ML quiz".

It converts inflected forms into the base form.

Stemming: has a similar purpose (reduce words by chopping them)

Say "automates", "automatic" could be reduced to "automat" which might not be an english word

We use spaCy for average words embeddings and other NLP stuff (words similarity etc) (spaCy is based on dense word embeddings as opposed to the sparse representation of bags of words)

## **Topic modeling pipeline**[**¶**](https://ubc-cs.github.io/cpsc330/lectures/16_natural-language-processing.html#topic-modeling-pipeline)

* Preprocess your corpus. (spaCy as well)
  + **Preprocessing is crucial!**
  + Tokenization, converting text to lower case
  + Removing punctuation and stopwords
  + Discarding words with length < threshold or word frequency < threshold
  + Possibly lemmatization: Consider the lemmas instead of inflected forms.
  + Depending upon your application, restrict to specific part of speech;
    - For example, only consider nouns, verbs, and adjectives
* Train LDA using Gensim.
* Interpret your topics.

Basic Text Processing

### Tokenization[¶](https://ubc-cs.github.io/cpsc330/lectures/16_natural-language-processing.html#tokenization)

* Sentence segmentation
  + Split text into sentences
* Word tokenization
  + Split sentences into words
* In English, period (.) is quite ambiguous. (In Chinese, it is unambiguous.)
  + Abbreviations like Dr., U.S., Inc.
  + Numbers like 60.44%, 0.98
* ! and ? are relatively ambiguous.
* How about writing regular expressions?
* A common way is using off-the-shelf models for sentence segmentation.
* Usually in NLP, we talk about
  + **Type** an element in the vocabulary
  + **Token** an instance of that type in running text

### Punctuation and stopword removal

* The most frequently occurring words in English are not very useful in many NLP tasks.
  + Example: the , is , a , and punctuation
* Probably not very informative in many tasks

### Lemmatization

* For many NLP tasks (e.g., web search) we want to ignore morphological differences between words
  + Example: If your search term is “studying for ML quiz” you might want to include pages containing “tips to study for an ML quiz” or “here is how I studied for my ML quiz”
* Lemmatization converts inflected forms into the base form.

### Stemming[¶](https://ubc-cs.github.io/cpsc330/lectures/16_natural-language-processing.html#stemming)

* Has a similar purpose but it is a crude chopping of affixes
  + automates, automatic, automation all reduced to automat.
* Usually these reduced forms (stems) are not actual words themselves.
* A popular stemming algorithm for English is PorterStemmer.
* Beware that it can be aggressive sometimes.