Computational Lexical Semantics

[J&M'00 Ch. 19]

Motivation:

Suppose our system is fed the following information:
FACT: *Some men abhor guns.*

And then, the following query:
QUERY: *Are there some people who hate pistols?*

ANSWER: *Yes*.

Why is it possible to provide this answer? The words don't match.

Computational Lexical Semantics

MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

Consisting of advanced components, mainframes have the capability of running multiple large applications required by many and most enterprises and organizations. This is one of its advantages. Mainframes are also suitable to cater for those applications (programs) or files that are of very high demand by its users (clients). Examples of such organizations and enterprises using mainframes are online shopping websites such as Ebay, Amazon, and computing-giant

MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple large applications required by most enterprises, which is one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very large demand by its users (clients). Examples of these include the large online shopping websites -i.e. : Ebay, Amazon, Microsoft, etc.

Computing lexical relations and lexical similarity:

- Plagiarism detection
- Web search engines
  (bag of words method, used by most search engines)
- Question answering
  (vector-based semantics)
- Automatic summarization

Synonymy: when different words have the same sense (meaning)

purse / *handbag*

sofa / *couch*

water / H_2O

big / *large*

remember / *recall*

purchase / *buy*

automobile / *car*

vomit / *throw up*

propose / *pop the question*

cannabis / *marijuana* / *weed* / *pot* / *grass* / *Mary Jane*

Antonymy: words with opposite meanings.

- **Complementary Antonymy**: complete opposites.
  (something is either one or the other, or neither)
  alive/dead; *married/unmarried*; *win/lose*; *leader/follower*.
- **Gradable Antonyms**: varying degrees of opposition.
  wet/dry; *easy/hard*; *poor/rich*; *love/hate*.
- **Reverses**: opposite trajectories of motion (not degree)
  rise/fall; *expand/contract*; *ascend/descend*.
- **Converses**: opposite perspectives of the SAME situation
  lend/borrow; *buy/sell*; *send/receive*; *employer/employee*.

- **Hyponymy**: a hierarchical relation between word meanings (X is a hyponym of Y if X is a more specific concept than Y)

  Hyponyms of *dog*:
  poodle, *bulldog*, *saint-bernard*, *pitbull*, ..., *lap dog*, *police dog*, *guide dog*, ...

  Hyponyms of *flower*:
  rose, *tulip*, *daisy*, *orchid*, *lily*, *lotus*, *gardenia*, *sunflower* ...

  Hyponyms of *jump*:
  leapfrog, hurdle, somersault, ...

Computational Lexical Semantics

← Hyponyms ↓

- primate
  - simian
    - ape
      - orangutan
      - gorilla
        - silverback
      - chimpanzee
    - monkey
      - baboon
      - macaque
      - vervet monkey
  - hominid
    - homo erectus
    - homo habilis
    - Neandertal man
    - homo sapiens
      - Cro-magnon
      - homo sapiens sapiens

↑ Hypernyms ↑

WordNet: a lexical database containing

- Synonymy ('synsets' = sets of synonyms, ordered by frequency in SemCor)
- Antonymy
- Hypernymy/Hyponymy
- Meronymy/Holonymy

82,115 nouns, 13,767 verbs, 18,156 adjectives, 3,621 adverbs

Freely available

- Online interface: `http://wordnetweb.princeton.edu/perl/webwn`
- For download: Prolog or NLTK

Wordnets for other languages

- EuroWordNet covers 7 European languages
  http://www.illc.uva.nl/EuroWordNet/
- GermaNet
  http://www.sfs.uni-tuebingen.de/lsd/
- Japanese WordNet
  http://nlpwww.nict.go.jp/wn-ja/index.en.html
- A global forum to coordinate them all
  http://www.globalwordnet.org

There are various other similar lexical databases

MeSH ONTOLOGY

- MeSH = Medical Subject Headings
- Machine-readable database of 250K terms
- Used for indexing 18M articles in MEDLINE
- Synonymy (cancer/tumor)
- Hyponyms (melatonin/hormone)
- Used in much BioNLP work

- All MeSH Categories
  - Diseases Category
    - Neoplasms
      - Neoplasms by Site
        - **Digestive System Neoplasms**
          - Biliary Tract Neoplasms
            - Bile Duct Neoplasms +
            - Gallbladder Neoplasms
          - Gastrointestinal Neoplasms
            - Esophageal Neoplasms
            - Gastrointestinal Stromal Tumors
            - Intestinal Neoplasms +
            - Stomach Neoplasms
          - Liver Neoplasms
            - Adenoma, Liver Cell
            - Carcinoma, Hepatocellular
            - Liver Neoplasms, Experimental
          - Pancreatic Neoplasms
            - Adenoma, Islet Cell +
            - Carcinoma, Islet Cell +
            - Carcinoma, Pancreatic Ductal
          - Peritoneal Neoplasms

Others

- Cyc
  (630.000+ concepts, 7,000,000 assertions, 38,000 relations)
- OpenCyc (open-source; 6.000 concepts)
- Semantic Web: Web Ontology Language
  `http://www.w3.org/2004/OWL/`

Hyponymy/Hypernymy are relevant for **entailment**.
X entails Y if whenever X is true, Y is also true

Downward-entailing verbs:

(1) a. I hate apes.
entails:
b. I hate orangutans.
c. I hate baboons. ...
d. I hate [*any-hyponym-of-ape*].
but does not entail:
e. I hate primates.
f. I hate animals.
g. I hate everything.

Entailment works the same way for any kind of noun:

(2) a. I hate cars.
   entails:
   b. I hate Volvos.
   c. I hate Hondas.
   d. I hate patrol cars.
   e. . . .
   but does not entail:
   f. I hate vehicles.
   g. I hate things.

Other downward-entailing verbs: *dislike*, *hate*, *despise*, *abhor*, *prohibit*, *fear*, ...

Upward-entailing verbs:

(3) a. I saw cars.
entails:
b. I saw vehicles.
c. I saw things.

but does not entail:
d. I saw Ferraris.
e. I saw dragsters.
f. I saw patrol cars.

Other upward-entailing verbs: *buy*, *have*, *read*, *write*, *mention*, . . .

Some quantifiers also have similar entailments:

- *exists*$(\uparrow,\uparrow)$

  (4) a. A cat yawned $\Rightarrow$
  A feline yawned.
  b. A cat yawned $\not\Rightarrow$
  A siamese cat yawned

  (5) a. Robin petted a black cat this morning $\Rightarrow$
  Someone touched an animal today
  b. Someone touched an animal today $\not\Rightarrow$
  Robin petted a black rat this morning

- *forall*$(\downarrow,\uparrow)$

  (6) a. Every kid petted a cat $\Rightarrow$ Every boy touched an animal
  b. Every boy touched an animal $\not\Rightarrow$ Every kid petted a cat

- *no*$(\downarrow,\downarrow)$

  (7) No cat ate a rodent today $\Rightarrow$
  No white cat devoured a black rat this morning

Computational Lexical Semantics

But in order to do anything useful, we need to be able to determine which sense of the word is relevant.

1. *bow*
    1. .. rod used to play certain string instruments
    2. ... the action of bending forward at the waist
    3. ... the front of the ship
    4. ... a weapon that shoots arrows
    5. ... a type of tied ribbon
    6. ...
2. *date*
    1. ... a fruit
    2. ... a time in the calendar
    3. ... a social meeting/appointment
    4. ...

Word Sense Disambiguation (WSD)

- **The KISS approach**: just select the most frequent sense. This yields 60% - 70% accuracy.
- **Supervised ML approach**: use hand-labeled corpora (called semantic concordances)
  - SemCor: 200K words from Brown with WordNet senses
  - SensEval-3: 2,081 tagged content words (Brown & WSJ)) SemEval-7: 6,000 words (http://www.senseval.org/)
  - DSO: 192K sentences from Brown & WSJ

  Steps:

  1. Pick a sense-annotated training corpus
  2. Extract features describing contexts of target word
  3. Train a classifier using some machine learning algorithm
  4. Apply classifier to unlabeled data

Extracting features that are predictive of word senses:

An electric guitar and bass player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps.

Collocational features	
word_L3	electric
POS_L3	JJ
word_L2	guitar
POS_L2	NN
word_L1	and
POS_L1	CC
word_R1	player
POS_R1	NN
word_R2	stand
POS_R2	VB
word_R3	off
POS_R3	RB

Bag-of-words features	
fishing	0
big	0
sound	0
player	1
fly	0
rod	0
pound	0
double	0
runs	0
playing	0
guitar	1
band	0

Naive Bayes: chooses the most likely sense s for a word w given j features of the context f.

$\hat{s} = argmax_{\mathsf{s} \in S} \; P(s) \prod_{j=1}^{n} P(f_j|s)$

$$P(s) = \frac{count(s, w)}{count(w)}$$

$$P(f_j|s) = \frac{count(f_j, s)}{count(s)}$$

- Usually work in Logspace
- Laplace smoothing is very common

What is '**Naive Bayes**' ?

If we apply Bayes' rule to this problem we would have to solve:

$$P(s|f_1...f_n) = \frac{P(f_1...f_n|s)P(s)}{P(f_1...f_n)}$$

But by assuming that the $f_1...f_n$ are independent (and their relative order does not matter), we can simplify:

$$P(s|f_1...f_n) = \frac{P(s)\prod_{j=1}^{n} P(f_j|s)}{P(f_1...f_n)}$$

And because the denominator is constant we remove it:

$$P(s|f_1...f_n) \approx P(s)\prod_{j=1}^{n} P(f_j|s)$$

Naive Bayes is widely used for classification, e.g. spam detection:

$$P(spam|w_1...w_n) \approx P(spam)\prod_{j=1}^{n} P(w_j|spam)$$

For $w_1...w_n$ words that are likely to appear in spam emails.
As well as in text classification:

$$P(class_5|w_1...w_n) \approx P(class_5)\prod_{j=1}^{n} P(w_j|class_5)$$

For $w_1...w_n$ (non-stop) words in a document

Dictionary-based methods to WSD

Simplified Lesk (1986)

- Retrieve all sense definitions of target word from dictionary
- Count words in context that overlap with definitions
- Choose the sense with the most overlapping words

Example:

The **cones** *from this* **pine** *tree are very small*

1. *pine*
   1. a kind of evergreen <u>tree</u> with needle-shaped leaves
   2. to waste away through sorrow or illness
2. *cone*
   1. a solid body which narrows to a point
   2. something of this shape, whether solid or hollow
   3. fruit of certain evergreen <u>tree</u>s
3. Disambiguation: $pine_1$ $cone_3$

A related task is **Word Similarity**

Some methods for word similarity focus strictly on degrees of similarity, but others extend to relatedness.

1. **Similarity**: near-synonymy
   sofa and *couch* are similar *cat feline* are somewhat
2. **Relatedness**
   car and *gasoline* are related, not similar

Applications

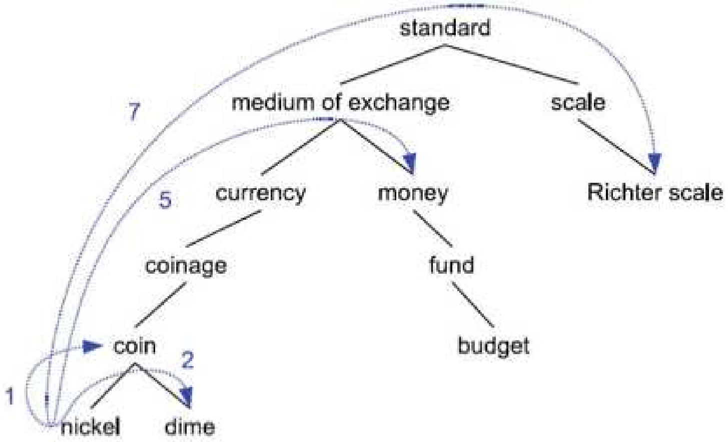
- Information retrieval
- Question answering
- Summarization
- Machine Translation
- Automatic essay grading
- Plagiarism detection

Various classes of word similarity methods

- Thesaurus-based (i.e. ontology-based)
  Are words 'nearby' in hypernym hierarchy?
  Do words have similar definitions (aka 'glosses')?
  E.g. **Extendend Lesk**
- Distributional algorithms
  Do words occur in the same environments?
- Hybrid
  E.g. **Corpus Lesk**

Path based similarity

Two senses are similar if they are hierarchically near each other.
For example, using WordNet:



Refinements

- **Path-length:**
  $pathlen(c_1, c_2) = 1 +$ number of edges in the shortest path in the hypernym graph between sense nodes c_1 and c_2

  Assumes each edge represents a uniform sense distance.

- **Path-based length similarity:**

  $sim_{path}(c_1, c_2) = \frac{1}{pathlen(c_1,c_2)}$

  and sometimes

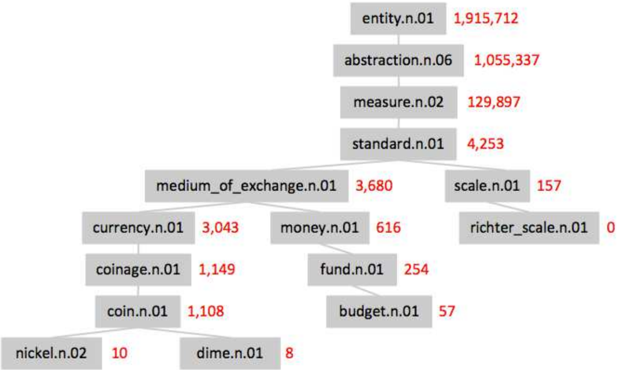
  $sim_{path}(c_1, c_2) = -log\, pathlen(c_1, c_2)$

- **Word similarity**
  $wordsim(w_1, w_2) = max\, sim(c_1, c_2)$ where $c_1 \in sense(w_1)$ and $c_2 \in sense(w_2)$

Information content similarity metrics
allow different degrees of similarity between senses/concepts.

nickel/money seem more related to each other than *nickel/standard*

Use sense counts to allow degrees of similarity



Information content similarity metrics

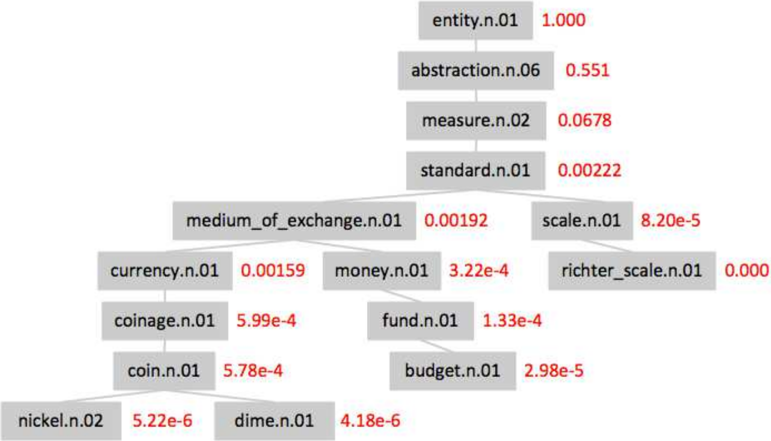
$P(entity) = 1$

$P(c)$ = the probability that a randomly selected word in a corpus is an instance of a concept c

$$P(c) = \frac{\sum_{w \in words(c)} count(w)}{N}$$

$words(c)$ = set of all words that are an instance of c
$count(w)$ = count occurrences of w in a corpus

One can then add the $P(c)$ to each concept c:



Information content:
$IC(c) = -logP(c)$
Lowest common subsumer:
$LCS(c_1, c_2) =$ lowest node that subsumes c_1 and c_2

- Resnik similarity metric:
$sim_{resnik}(c_1, c_2) = -logP(LCS(c_1, c_2))$
$sim_{resnik}(nickel, money) = -logP(medium\_of\_exchange) = 9.02$
$sim_{resnik}(nickel, standard) = -logP(standard) = 8.81$

- Lin (Resnik) similarity

$$sim_{Lin}(c_1, c_2) = \frac{2 \times logP(LCS(c_1, c_2))}{logP(c_1) + logP(c_2)}$$

$$sim_{Lin}(nickel, money) = \frac{2 \times logP(medium\_of\_exchange)}{logP(nickel) + logP(money)} = \frac{2 \times -9.02}{-14.6 - 6.79} = 0.843$$

$$sim_{Lin}(nickel, standard) = \frac{2 \times logP(standard)}{logP(nickel) + logP(money)} = \frac{2 \times -9.02}{-14.6 - 8.81} = 0.752$$

- Jiang-Conrath distance is similar to Lin's but instead describes the distance between two concepts:

$$dist_{JC}(c_1, c_2) = 2 \times logP(LCS(c_1, c_2)) - (logP(c_1) + logP(c_2))$$

The similarity can be derived by computing the reciprocal:

$$sim_{JC}(c_1, c_2) = \frac{1}{2 \times logP(LCS(c_1, c_2)) - (logP(c_1) + logP(c_2))}$$

Tools to compute word similarity:

1. NLTK
2. WordNet::Similarity

Lesk word similarity

Two concepts are similar if their glosses contain similar words.

(8) a. drawing paper: *paper that is specially prepared for use in drafting*

b. decal: *the art of transferring designs from specially prepared paper to a wood or glass or metal surface*

For each n-word sequence occurring in both glosses, add a score of n^2
Example: 'paper' & 'specially prepared' $= 1^2 + 2^2 = 5$

Extended Lesk: do the same for other relations (glosses of hypernyms and hyponyms)

$$sim_{eLesk}(c_1, c_2) = \sum_{r,q \in RELS} overlap(gloss(r(c_1)), gloss(q(c_2)))$$

Computing word similarity without databases (thesauri)

The problems of using databases, like dictionaries or ontologies like WordNet:

- Some word senses are missing
- Some connections between senses are missing
- Work less well for adjectives and verbs: have less structured hyper/hyponymy networks

Intuition for distributional word similarity approaches:

> *A bottle of tesgüino is on the table. Everybody likes tesgüino. It makes you drunk. We make tesgüino out of corn.*

You would guess that ‘tesgüino’ must be some alcoholic beverage.

Intuition: words are similar if they have similar word contexts.

Vector-space models of meaning

Word/Term-context matrix for word similarity: collect and count the words that occur up to 10 words to the left of the word under evaluation and up to 10 words to the right.

Example: comparing w and w'

$w_0 \ldots w_9 \boxed{w} w_{10} \ldots w_{20}$

$w'_0 \ldots w'_9 \boxed{w'} w'_{10} \ldots w'_{20}$

The shorter the window the more syntactiy (1-3 very!), the longer the more semanticy (4-10).

Comparing the 4 words in the left column:

	aardvark	computer	data	pinch	result	sugar	…
apricot	0	0	0	1	0	1	
pineaple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

For a vocab of 50k words, the matrix is 50K $\times$ 50k and very sparse.

Various kinds of vector models

1. Sparse vector representations
   - Mutual-information weighted word co-occurrence matrices
2. Dense vectors representations
   - Singular value decomposition (and Latent Semantic Analysis)
   - Neural-network-based models (word2vec, CBOW, Skip-Gram, Glove)
   - Brown clusters

For sparse word occurrence matrices:
Pointwise Mutual Information (from $-\infty$ to $+\infty$)

Intuition: do x and y occur more than if they were independent?

$$PMI(x,y) = log_2 \frac{P(x,y)}{P(x) \times P(y)}$$

Positive PMI: negative values are replaced with 0 (what does 'negatively related' even mean?).

$$PPMI(x,y) = max\left(log_2 \frac{P(x,y)}{P(x) \times P(y)}, 0\right)$$

Positive Pointwise Mutual Information

f_{ij} = number of w_i co-occurrences with (context) word c$_j$

- Probability of a word w_i occurring with c_j

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W} \sum_{j=1}^{C} f_{ij}}$$

- Probability of a word w_i occurring with all its c's

$$p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W} \sum_{j=1}^{C} f_{ij}}$$

- Probability of a context

$$p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W} \sum_{j=1}^{C} f_{ij}}$$

- PMI $\quad pmi_{ij} = log_2 \frac{p_{ij}}{p_{i*} p_{*j}}$
- Positive PMI

$$ppmi_{ij} = \left\{ \begin{array}{l} pmi_{ij} \text{ if } pmi_{ij} > 0 \\ 0 \text{ otherwise} \end{array} \right\}$$

Example

	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineaple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

Solving for: $p(i = information, j = data)$

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} = \frac{6}{19} = .31$$

$$p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} = \frac{11}{19} = .57$$

$$p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} = \frac{7}{19} = .36$$

$$pmi(ij) = log_2 \frac{p_{ij}}{p_{i*}p_{*j}} = log_2(\frac{.31}{.57 \times .36}) = .59$$

Applied to other words the general outcome is:

	computer	data	pinch	result	sugar
apricot	–	–	2.25	–	2.55
pineaple	–	–	2.25	–	2.55
digital	1.66	0.00	–	0.00	–
information	0.00	0.58	–	0.47	–

'–' $=$ negative PPMI to be replaced with 0's

However, PMI is biased toward infrequent events (very rare words have very high PMI values). One can give rare words slightly higher probabilities (Laplace smoothing).

Another vector similarity measure is:

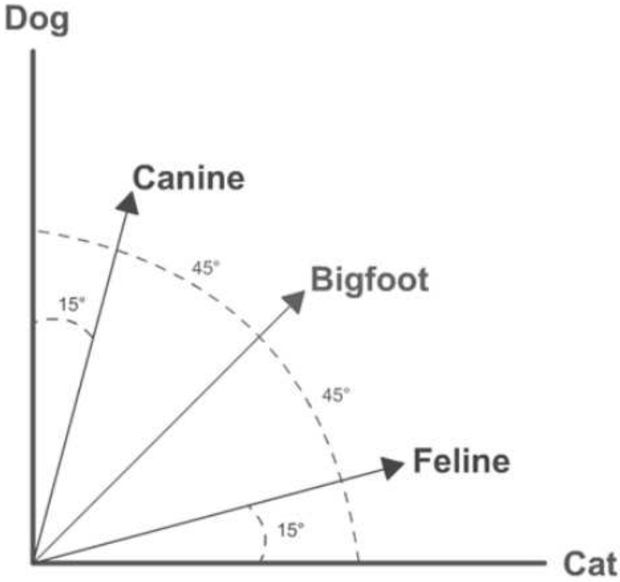
$$distance_{Euclidean}(\vec{q}, \vec{d}) = \sqrt{\sum_{i=1}^{N} (q_i - d_i)^2}$$

... Sensitive to the vector length. Better metrics are:

$$sim_{Jaccard}(\vec{q}, \vec{d}) = \frac{\sum_{i=1}^{N} min(q_i, d_i)}{\sum_{i=1}^{N} max(q_i, d_i)}$$

$$similarity_{cosine}(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \times |\vec{y}|} = \frac{\sum_{i=1}^{n} x_i \times y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \times \sqrt{\sum_{i=1}^{n} y_i^2}}$$

- For example, the dot product $[0, 0, 0, 1, 0, 1] \cdot [0, 2, 1, 0, 1, 0] = (0 \times 0) + (0 \times 2) + (0 \times 1) + (1 \times 0) + (0 \times 1) + (0 \times 0) = 0$
- (L2) Norm of a vector $\vec{x}$
  $|\vec{x}| = \sqrt{\vec{x} \cdot \vec{x}}$


Dog
Canine
45°
Bigfoot
15°
45°
Feline
15°
Cat

There are many vector compression techniques. A particularly simple and efficient method is **Random Indexing**

1. Create a random vector for each of the words that appear in the 10-word long window (all 0', one 1, and one -1):

w_1	[1,	0,	-1]
w_2	[0,	1,	-1]
w_3	[0,	-1,	1]
w_4	[-1,	1,	0]

2. Initialize word context vector for the target words as all 0's:

t_1	[0,	0,	0]
t_2	[0,	0,	0]

3. Every time we encounter w_n near the target word t_m, add w_n's vector to t_m's vector. Suppose w_1 is found near t_1 once. Then we get:

t_1	[1,	0,	-1]
t_2	[0,	0,	0]