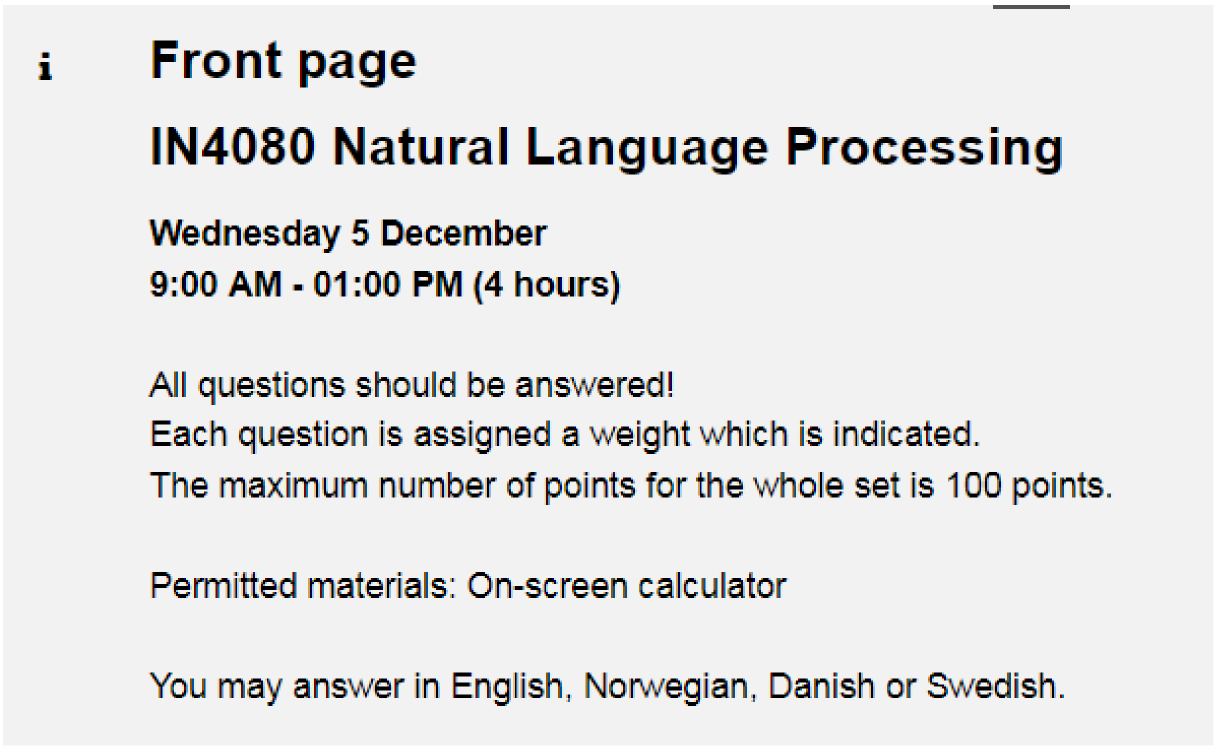

i

# Front page

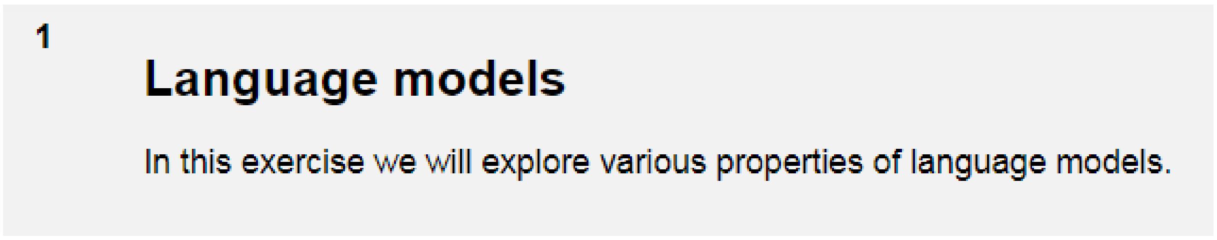
# IN4080 Natural Language Processing

**Wednesday 5 December**
**9:00 AM - 01:00 PM (4 hours)**

All questions should be answered!
Each question is assigned a weight which is indicated.
The maximum number of points for the whole set is 100 points.

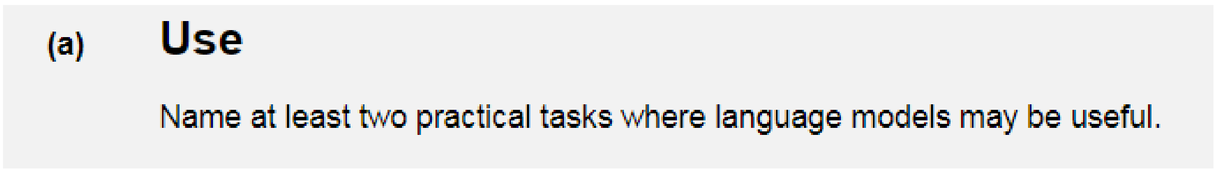
Permitted materials: On-screen calculator

You may answer in English, Norwegian, Danish or Swedish.


1

## Language models

In this exercise we will explore various properties of language models.


(a)

### Use

Name at least two practical tasks where language models may be useful.

Maks. poeng: 4

Any two of the following 6 (or others) will do:

1. One can use one language model and compare the probability of two or more sentences given the model. This may be applied to
1.1 Speech recognition
1.2 Machine translation
1.3 Context-dependent spelling correction

2. One can use two or more language models and compare the models with respect to a text corpus asking which model makes the corpus most probable. This may be applied e.g. in
2.1 Language identification
2.2 Author attribution
2.3 Genre classification

**(b)** **Model**

Explain how a bigram language model will assign a probability to a sequence of words. You are advised to use (simplified) equations in your explanation. Which assumption is made by the model?

Maks poeng: 6

We will write $w_1^n$ for the token sequence $w_1, w_2, \dots w_n$
Then the probability of the token sequence is

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k | w_1^{k-1})$$

according to the chain rule.

A bigram language model makes the further assumption that
$P\left(w_k \middle| w_1^{k-1}\right) = P\left(w_k \middle| w_{k-1}\right)$ for k$\geq$2, i,e, that the probability of a token only depends on its immediate predecessor and not on the whole history, and hence that

$$P(w_1^n) = \prod_{i=1}^{n} P(w_k | w_1^{k-1}) = P(w_1) \prod_{k=2}^{n} P(w_k | w_{k-1})$$

**i** **Corpus 1**

**Corpus 1**

<s> Sam likes Pam <\s>
<s> Pam likes Sam <\s>
<s> Sam likes egg and ham<\s>

**(c)** **Bigram probability**

Consider a language model trained on corpus 1. Which conditional bigram probability will it ascribe to P(likes | Sam), assuming a straightforard unsmoothed maximum likelikehood estimation?

Maks. poeng: 2

$$P(likes\ |Sam) = \frac{count(Sam\ likes)}{count(Sam)} = \frac{2}{3}$$

**(d)** **Sentence probability**

Which probability will the model ascribe to the following sequence?
<s> Sam likes Sam <\s>
Show how the number is calculated.

Maks. poeng: 4

$$P(\langle s \rangle\ Sam\ likes\ Sam\ \langle \backslash s \rangle) =$$
$$P(\langle s \rangle) P(Sam \mid \langle s \rangle) P(likes \mid Sam) P(Sam \mid likes) P(\langle \backslash s \rangle \mid Sam) =$$
$$1 * \frac{2}{3} * \frac{2}{3} * \frac{1}{3} * \frac{1}{3} = \frac{4}{81} = 0.49$$

**(e) Problem**

Which problems will the bigram model face in assigning a probability to the following sentence?
<s> Sam likes ham <\s>

Maks. poeng: 2

*P(ham | likes)* = 0 since 'likes ham' does not occur in the corpus, hence the whole sentence gets probability 0.

**(f) Interpolation**

One way of fixing these problems is to use interpolation. Explain how interpolation works for a bigram model.

Maks. poeng: 4

Interpolation works by combining an n-gram model with n-gram models with shorter n-grams. For a bigram model that means to combine the bigram model with a unigram model and replacing $P\left(w_k \middle| w_{k-1}\right)$ with
$P'\left(w_k \middle| w_{k-1}\right) = l_2 * P\left(w_k \middle| w_{k-1}\right) + l_1 * \mathrm{P}(w_k)$, for some suitable choice of $l_1$ and $l_2$where $l_1 + l_2 = 1$. Normally, $l_1$ and $l_2$ are determined by a maximal likelihood on a held out set.

**(g) Applying interpolation**

Which probability will an interpolated bigram model ascribe to the sentence?
<s> Sam likes ham <\s>
You may assume a 0.5-0.5 weighting.
Show how you get the result.

Maks. poeng: 4

$$
\begin{aligned}
&P'(\langle s\rangle\ Sam\ likes\ ham\ \langle \backslash s\rangle) = \\
&P(\langle s\rangle)P'(Sam \mid \langle s\rangle)P'(likes \mid Sam)P'(Sam \mid likes)P(\langle \backslash s \mid ham\rangle) = \\
&\qquad P(\langle s\rangle)\big(0.5 * P(Sam \mid \langle s\rangle) + 0.5 * P(Sam)\big) * \\
&\qquad (0.5 * P(likes \mid Sam) + 0.5 * P(likes)) * \\
&\qquad \big(0.5 * P(ham \mid likes) + \ 0.5 * P(ham)\big) * \\
&\qquad (0.5 * P(\langle \backslash s \mid ham\rangle) + 0.5 * P(\langle \backslash s\rangle) =
\end{aligned}
$$

$1 * 0.5 * \left(\frac{2}{3} + \frac{3}{17}\right) * 0.5 * \left(\frac{2}{3} + \frac{3}{17}\right) * 0.5 * \left(0 + \frac{1}{17}\right) * 0.5 * \left(1 + \frac{3}{17}\right) = \frac{0.5^4 * 43^2 * 1 * 20}{51^2 * 17^2} =$
0.0031

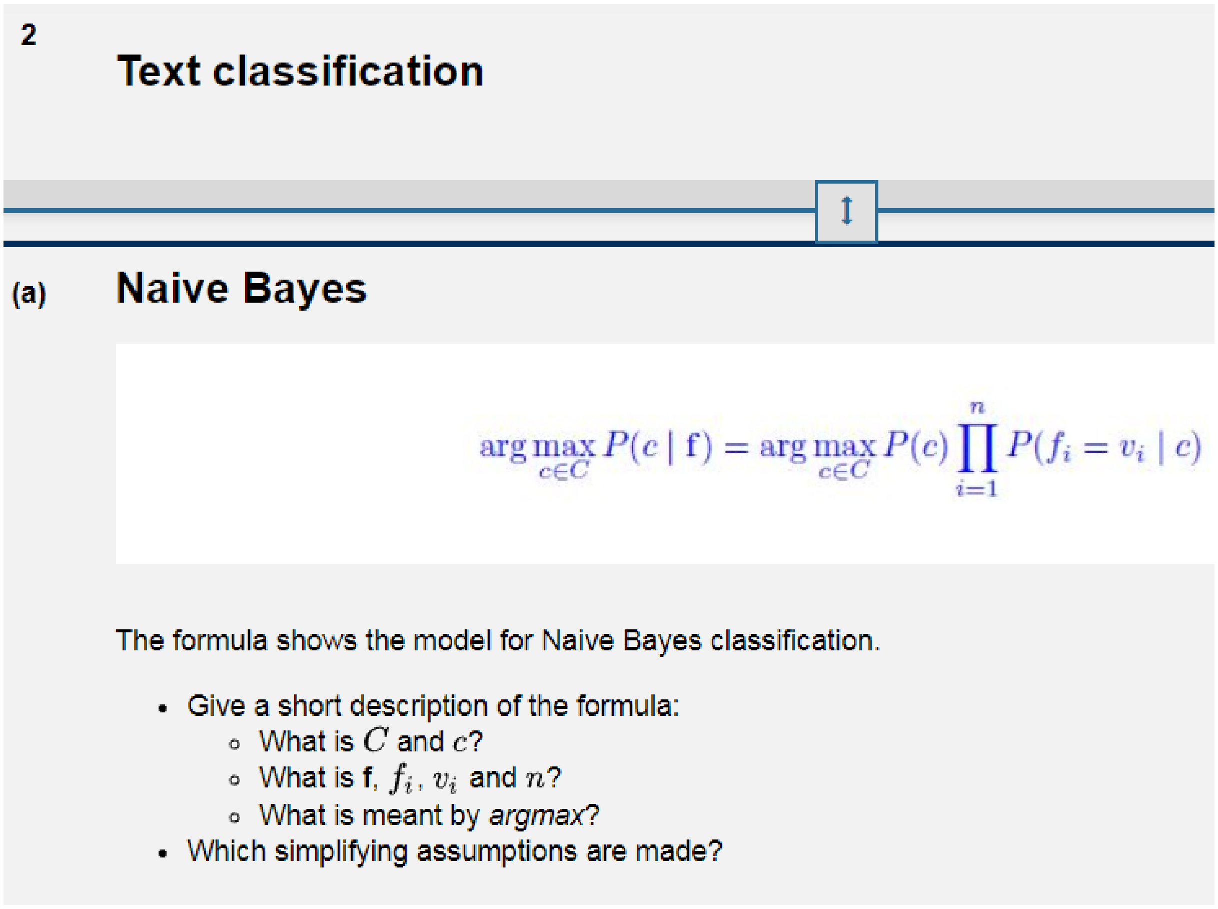
**(h) Add one**

In general, it is not a good idea to use Laplace smoothing ("add-one") to smooth bigram probabilities. Explain why!

Maks. poeng: 3

To use Laplace smoothing to bigrams will move too much probability mass to events we have never seen and will never se. For example, considering a corpus of all texts written by Shakespeare, it is roughly 900,000 word (tokens) and 30,000 different word types.

From the 30,000 word types there are 900 millions possible bigrams but we cannot have seen more than 900,000 of these. Using Laplace smoothing will roughly put 99.9% of the probability mass on unseen events and only use 0.1% for the observed bigrams.



## 2 Text classification

### (a) Naive Bayes

$$\arg\max_{c \in C} P(c \mid \mathrm{f}) = \arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(f_i = v_i \mid c)$$

The formula shows the model for Naive Bayes classification.

- Give a short description of the formula:
  - What is $C$ and $c$?
  - What is **f**, $f_i$, $v_i$ and $n$?
  - What is meant by *argmax*?
- Which simplifying assumptions are made?

Maks. poeng: 5

c indicates one class, C is the set of all classes.
$f_i$is one feature, $v_i$is the value of this feature
$n$ is the number of features
**f** is the feature vector $\boldsymbol{f} = \langle f_1 = v_1, f_2 = v_2, \dots, f_n = v_n \rangle$ which may also be written
$\boldsymbol{f} = \langle v_1, v_2, \dots, v_n \rangle$ if the order of features is determined
$P(f_i = v_i \mid c)$ is the probability that the feature $f_i$takes the value $v_i$given the class $c$
*argmax_{c in C}* means consider the expression within the scope of *argmax* for each *c* in *C* and choose the *c* that yields the largest value.

The simplifying assumption is that the value of each feature given a class is independent of the values of the other features, i,e., that

$$P(f_1 = v_1, f_2 = v_2, \dots f_n = v_n | c) = \prod_{k=1}^{n} P(\, f_k = v_k | c)$$

**i** **Data set**

**Data set:**

| Document | Class |
|---|---|
| fun, fun, good, good, bad | pos |
| exciting, good, exciting | pos |
| fun, exciting, exciting, bad | pos |
| bad | neg |
| terrible, terrible, terrible, terrible, terrible | neg |

## (b) Multinomial

There are several ways the Naive Bayes model can be used for text classification. One of them is called *Multinomial Naive Bayes*. Given the two classes *pos* and *neg*, and the data set, what are the values of the $P(f_i = v_i \mid c)$ for $i = 1, 2, \dots n$?

Maks. poeng: 5

| v | P(v \| pos) | P(v \| neg) |
|---|---|---|
| fun | 3/12 | 0 |
| good | 3/12 | 0 |
| bad | 2/12 | 1/6 |
| exciting | 4/12 | 0 |
| terrible | 0 | 5/6 |

## (c) Binarized NB

A variant of Multinomial Naive Bayes is called Binarized Multinomial Naive Bayes. We saw in mandatory assignment 2 how this binarized variant was potentially better for sentiment classfication. How does the binarized variant differ from the standard Multinomial NB? What are the values of the $P(f_i = v_i \mid c)$ for $i = 1, 2, \dots n$ in the the binarized model?

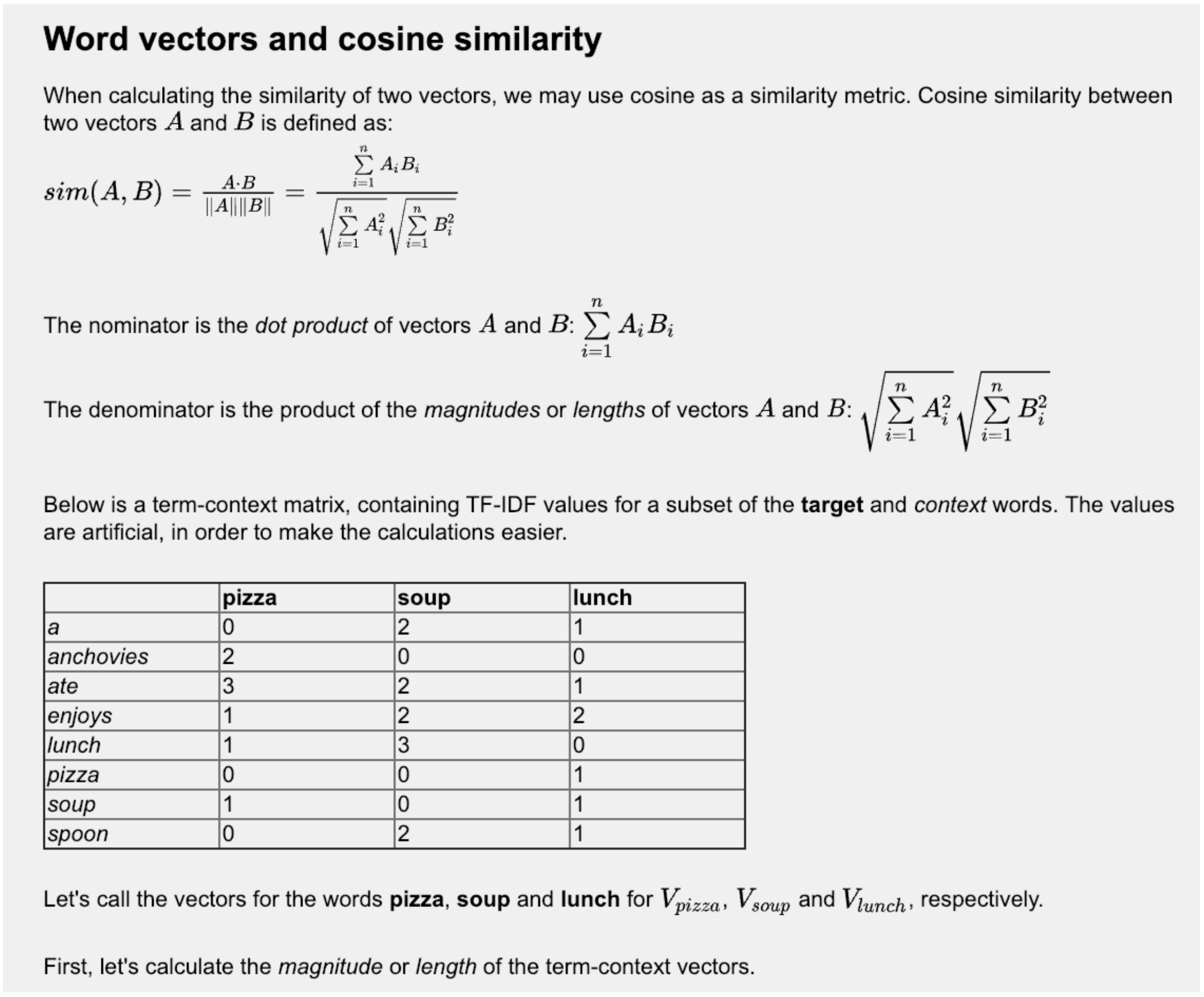
Maks. poeng: 5

A helping step

| **Document** | **Class** |
|---|---|
| fun, good, bad | pos |
| exciting, good | pos |
| fun, exciting, bad | pos |
| bad | neg |
| terrible | neg |

| v | P(v \| pos) | P(v \| neg) |
|---|---|---|
| fun | 2/8 | 0 |
| good | 2/8 | 0 |
| bad | 2/8 | 1/2 |
| exciting | 2/8 | 0 |
| terrible | 0 | 1/2 |

## Part 3: Text classification



### Word vectors and cosine similarity

When calculating the similarity of two vectors, we may use cosine as a similarity metric. Cosine similarity between two vectors $A$ and $B$ is defined as:

$$sim(A,B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

The nominator is the *dot product* of vectors $A$ and $B$: $\sum_{i=1}^{n} A_i B_i$

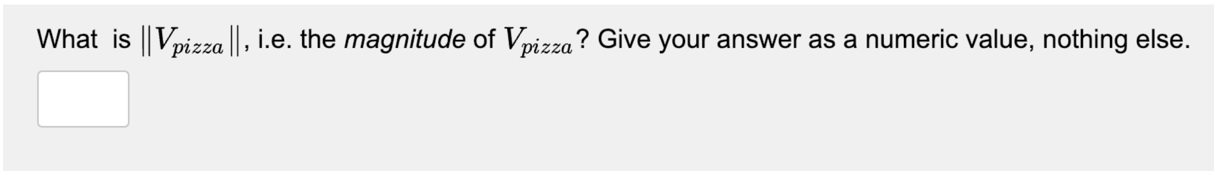
The denominator is the product of the *magnitudes* or *lengths* of vectors $A$ and $B$: $\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}$

Below is a term-context matrix, containing TF-IDF values for a subset of the **target** and *context* words. The values are artificial, in order to make the calculations easier.

| | **pizza** | **soup** | **lunch** |
|---|---|---|---|
| *a* | 0 | 2 | 1 |
| *anchovies* | 2 | 0 | 0 |
| *ate* | 3 | 2 | 1 |
| *enjoys* | 1 | 2 | 2 |
| *lunch* | 1 | 3 | 0 |
| *pizza* | 0 | 0 | 1 |
| *soup* | 1 | 0 | 1 |
| *spoon* | 0 | 2 | 1 |

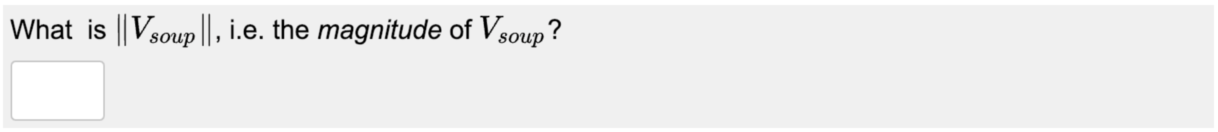
Let's call the vectors for the words **pizza**, **soup** and **lunch** for $V_{pizza}$, $V_{soup}$ and $V_{lunch}$, respectively.

First, let's calculate the *magnitude* or *length* of the term-context vectors.



What is $\|V_{pizza}\|$, i.e. the *magnitude* of $V_{pizza}$? Give your answer as a numeric value, nothing else.

$$\begin{aligned} &\sqrt{0^2+2^2+3^2+1^2+1^2+0^2+1^2+0^2} &&= \\ &\sqrt{0 \quad +4+9+1+1+0+1+0} &&= \\ &\sqrt{16} &&= \mathbf{4} \end{aligned}$$



What is $\|V_{soup}\|$, i.e. the *magnitude* of $V_{soup}$?

$$\begin{aligned} &\sqrt{2^2+0^2+2^2+2^2+3^2+0^2+0^2+2^2} &&= \\ &\sqrt{4+0+4+4+9+0+0+4} &&= \\ &\sqrt{25} &&= \mathbf{5} \end{aligned}$$

What is $||V_{lunch}||$, i.e. the *magnitude* of $V_{lunch}$?

$\sqrt{1^2+0^2+1^2+2^2+0^2+1^2+1^2+1^2} =$
$\sqrt{1+0+1+4+0+1+1+1} =$
$\sqrt{9} = \mathbf{3}$

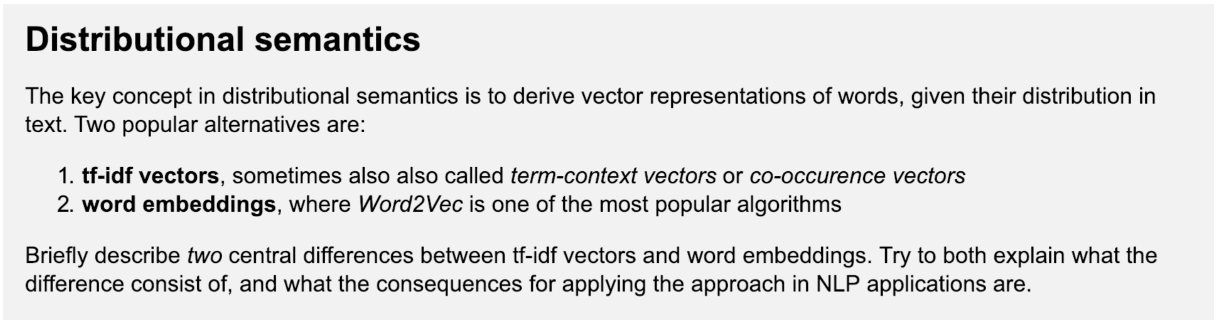
Using cosine similarity as a similarity metric, what is the similarity between the following word pairs?

**pizza** and **soup**

$$\frac{V_{pizza} \cdot V_{soup}}{||V_{pizza}||||V_{soup}||} =$$
$$\frac{0\times2+2\times0+3\times2+1\times2+1\times3+0\times0+1\times0+0\times2}{4\times5} =$$
$$\frac{0+0+6+2+3+0+0+0}{20} =$$
$$\frac{11}{20} = \mathbf{0.55}$$

**pizza** and **lunch**

$$\frac{V_{pizza} \cdot V_{lunch}}{||V_{pizza}||||V_{lunch}||} =$$
$$\frac{0\times1+2\times0+3\times1+1\times2+1\times0+0\times1+1\times1+0\times1}{4\times3} =$$
$$\frac{0+0+3+2+0+0+1+0}{12} =$$
$$\frac{6}{12} = \mathbf{0.5}$$


## Distributional semantics

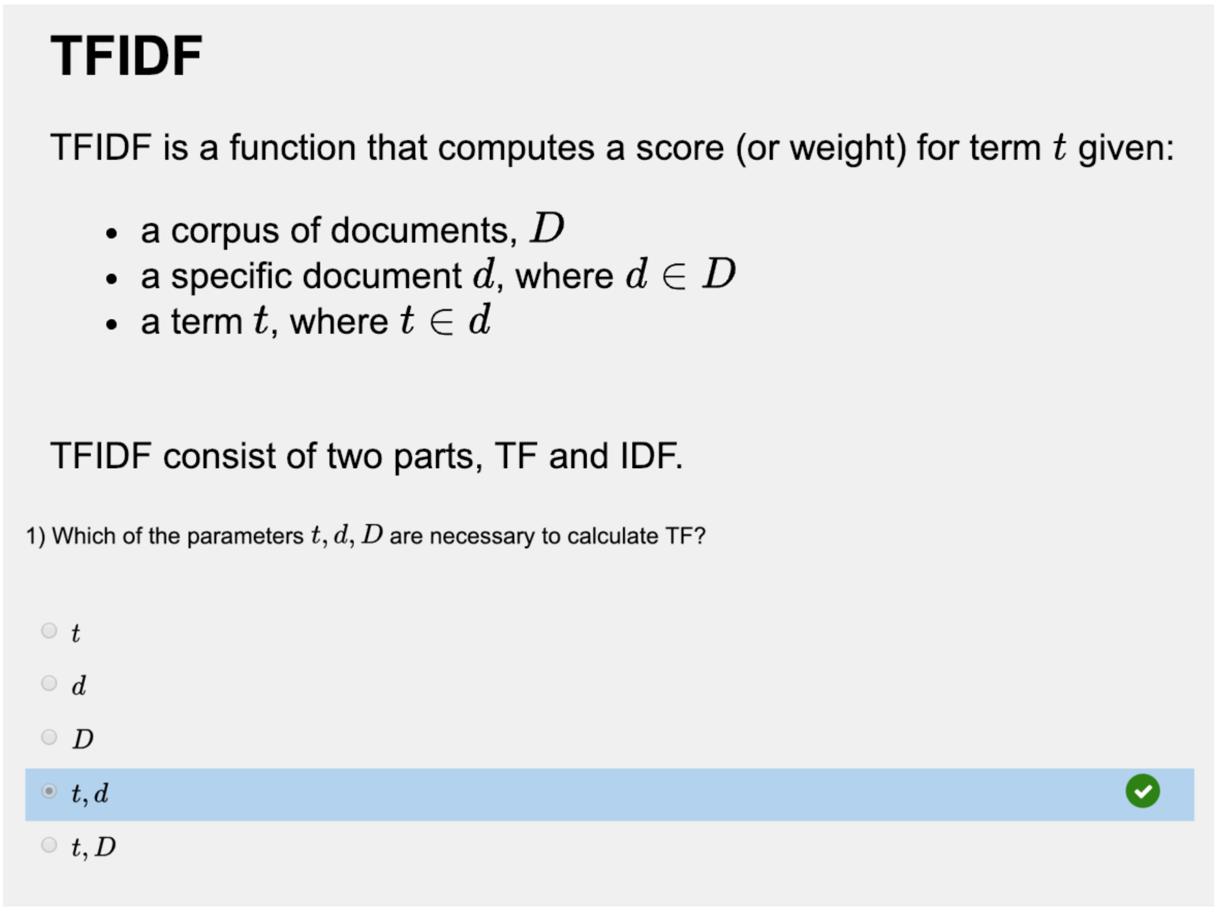
The key concept in distributional semantics is to derive vector representations of words, given their distribution in text. Two popular alternatives are:

1. **tf-idf vectors**, sometimes also also called *term-context vectors* or *co-occurence vectors*
2. **word embeddings**, where *Word2Vec* is one of the most popular algorithms

Briefly describe *two* central differences between tf-idf vectors and word embeddings. Try to both explain what the difference consist of, and what the consequences for applying the approach in NLP applications are.

Two central differences:

1. **tf-idf vectors** derive the word vectors from *counting* occurrences in the document, and then weighing the words by their inverse document frequencies (also *counting)*. **Word embeddings**, on the other hand, *learns* the word vectors by trying to *predict* the context of the word given the word, or by trying to *predict* the word given the context of the word
2. **tf-idf vectors** are *sparse*, i.e. they are of high dimensionality and contain tuypically many zeros. **Word embeddings** are dense, i.e. low dimensionality (typically hundreds of dimensions).


## TFIDF

TFIDF is a function that computes a score (or weight) for term $t$ given:

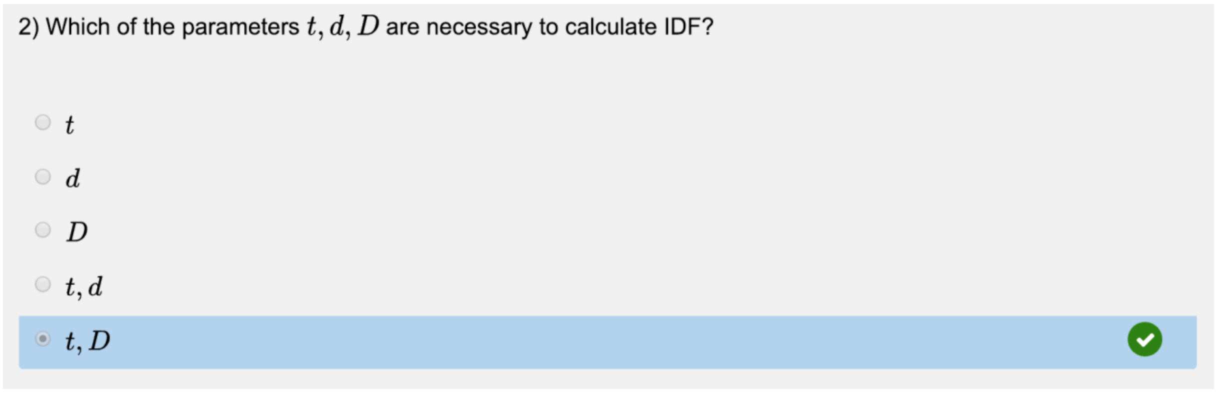
- a corpus of documents, $D$
- a specific document $d$, where $d \in D$
- a term $t$, where $t \in d$

TFIDF consist of two parts, TF and IDF.

1) Which of the parameters $t, d, D$ are necessary to calculate TF?

- $t$
- $d$
- $D$
- [✓ $t, d$]
- $t, D$

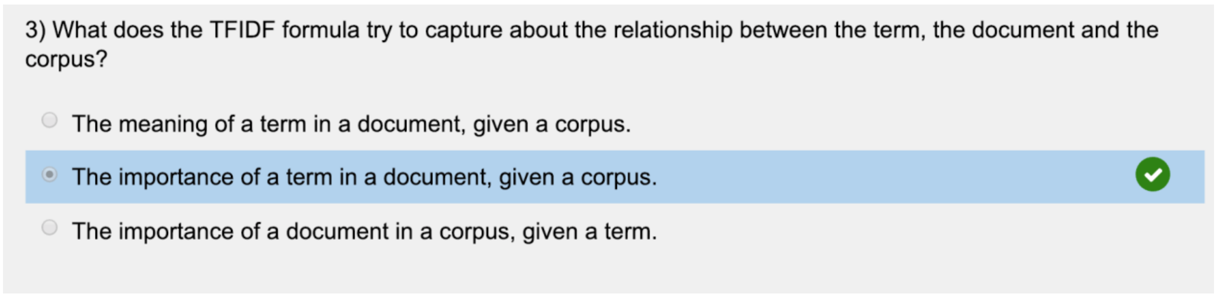
You need the term and the (relevant) document to calculate the value for TF.



2) Which of the parameters $t, d, D$ are necessary to calculate IDF?

- $t$
- $d$
- $D$
- $t, d$
- [✓ $t, D$]

You need the term and all the documents in the corpus to calculate the value for IDF.



3) What does the TFIDF formula try to capture about the relationship between the term, the document and the corpus?

- The meaning of a term in a document, given a corpus.
- [✓ The importance of a term in a document, given a corpus.]
- The importance of a document in a corpus, given a term.

TFIDF is used to weigh terms by their (estimated) importance. It does not express anything about the meaning of a term, even though it it used to *calculate* e.g. tf-idf vectors. Also, it does not say anything about the importance of a whole document.

4

## Dependency structure and dependency parsing

i

### Structure 1

Structure 1

| | word | lemma | upos | xpos | head | type |
|---|---|---|---|---|---|---|
| 1 | What | what | PRON | WP | 6 | obj |
| 2 | do | do | AUX | VBP | 4 | aux |
| 3 | you | you | PRON | PRP | 4 | nsubj |
| 4 | like | like | VERB | VB | 0 | root |
| 5 | to | to | PART | TO | 6 | mark |
| 6 | do | do | VERB | VB | 4 | xcomp |
| 7 | ? | ? | PUNCT | . | 4 | punct |

(a)

### Projectivity

This (simplified) dependency structure is taken from the UD_English treebank.

Is the structure projective or not?
State reasons for your answer.

Maks. poeng: 5

The structure is not projective

Definitions of projectivity

- Intuition: arcs don't cross each other
- A head A and dependent B must be adjacent:
  - every word between A and B must be subordinate to A
- If *i* --> j and (*i*<*k*<*j*) or (*j*<*k*<*i*) then *i* --> * *k*

Here 6 --> 1 but we do not have 6 -->* 4
Alternative explanation: The arc 6--> 1 crosses 4 --> 7 (and 4 --> 0)

(b)

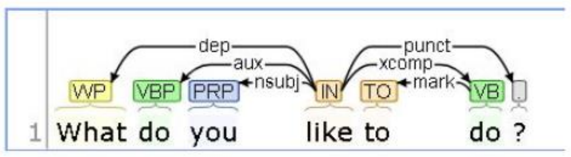
### Evaluation metrics

In evaluation of dependency structures, two metrics are often used: unlabeled attachment score (**UAS**) and labeled attachment score (**LAS**). Explain briefly these metrics and in particular the differences between them.

Maks. poeng: 4

Unlabelled attachment score counts how many tokens are ascribed the correct head
Labelled attachment score counts how many words are ascribed the correct head and the correct label.

(c) **LAS and UAS**



When the CoreNLP system parses the sentence, it produces the structure above, call it structure (2).

Taking structure(1) as the gold standard, what is the LAS and UAS of structure (2)?

Maks. poeng: 3

All words except 'What' is ascribed correct head and label; hence both LAS and UAS are 6/7. (Since the graph notation does not display the root edge while the conll format does, also 5/7 is considered correct here).

(d) **Transition-based dependency parsing**

Show step-by-step how a transition-based parser can produce structure (2). Be explicit with respect to the data structures used and the effect of each step.

Maks. poeng: 8

One has to choose between arc-standard or arc-eager. We show the arc-eager (malt-parser). We use the following format

stack | buffer - last move, dependency label added to structure, dependency relation added

0 | What do you like to do ? - start
0 What | do you like to do ? - shift
0 What do | you like to do ? - shift
0 What do you | like to do ?, shift
0 What do | like to do ?, LeftArc - 4:like --> 3:you, nsubj
0 What | like to do ? - LeftArc, 4:like --> 2:do, aux
0 | like to do ?, - LeftArc, 4:like --> 1:What, dep
0 like | to do ? - RightArc, 0 --> 4:like, root
0 like to | do ?- shift,
0 like | do ? - LeftArc, 6:do --> 5:to, mark
0 like do | ? - RightArc, 4:like --> 6:do, xcomp
0 like | ? - Reduce
0 like ? | - RightArc, 4:like --> 7:?, punct
0 like | - Reduce
0 | - Reduce

5

## Information extraction

(a) **Steps**

What are the typical steps of an information extraction system? Explain what the goals are for each step. You do not have to explain how the actual steps are carried out.

Maks. poeng: 5

The overall goal is to extract information from textual material.

Clean up
First, make sure that the data is interpreted correctly as text, taking into consideration e.g., encoding. May also be necessary to remove (or put aside) meta-data as e.g., XML- or HTML-tags.

Sentence segmentation
Split the text into units corresponding to sentences.

Word tokenization
Split each sentence into a sequence of smaller units corresponding to words. May also be units corresponding to punctuation.

Part of speech tagging.
For each sentence, tag the words with a part-of-speech tag, e.g., whether the word is a noun or a verb.

Chunking
Gather words into so-called NP-chunks, e.g.,
(The president) of the (United States) gave (a speech) .
These are maximally large NPs that do not contain other NP chunks.

Named entity recognition
To each NP chunk, decide whether it can name an entity and if it can, assign a class from a predefined small set of classes, e.g., person which may apply to "the president" in the example, and organization which might apply to "the United States". In a different context \the United States" could be classified as a location.

Relation extraction
Extract relations that exist between the named entities of the text. Normally a pre-defined set of relations determined by the purpose of the application, e.g., for medical records, this could include date of birth, has symptom, has diagnosis etc.

Additional steps
The extraction of temporal expressions and events can be additional steps which may be used to extract e.g., not only that the patient has a symptom, but when the symptom first appeared.

**(b) Manual method**

One of the steps in an information extraction system is relation extraction. There are several different methods for relation extraction. One method is to use hand-written patterns. Explain shortly the main principles of this approach. What are the bottlenecks of this approach?

Maks. poeng: 5

Hand written patterns
A person tries to identify patterns that are commonly used to express the relation, e.g.,
x wrote y,
x is the author of y
y is a book by x
for authorship.
The bottleneck is to write these patterns general enough, and to include sufficiently many of them, as there are often many ways to express the same.

**(c) Supervised**

An alternative method is to use supervised classification. Explain shortly the main principles of this method. What are the bottlenecks of this method?

Maks. poeng: 5

Supervised classification
Sentences are manually annotated with named entities and relations between them. This is used to train a classifier which can then be used to assign relations to other sentences (after they have been tagged, chunked and named-entity recognized).

The bottleneck is that one needs much training material to get this to work satisfactorily and that it is resource demanding to make this training material.