Discovering Collocations

Remember *collocations* from Informatics 1B?

- collocations are sequences of words that occur together;
- correspond to conventionalized, habitual ways of saying things;
- are often highly frequent in the language;
- collocations contrast with other expressions that are near-synonyms, but not conventionalized (*strong tea* vs. *powerful tea*; *strong car* vs. *powerful car*);

*Task*: automatically identify collocations in a large corpus.

(1) He spoke English with a/n ... French accent.

a. average  
b. careless  
c. widespread  
d. *pronounced*  
e. chronic
Discovering Collocations

(2) He gave us a ... account of all that you had achieved over there.
   a. ready
   b. yellow
   c. careless
   d. luxury
   e. glowing

(3) Could you please give me a/n ... account?
   a. itemized
   b. dreadful
   c. great
   d. luxury
   e. glowing

(4) Kim and Sandy made ... after the argument.
   a. with
   b. about
   c. off
   d. up
   e. for

Why do we care about collocations? In cognitive science:

- Speakers of a language have strong intuitions about collocations (see previous slides).
- Where do these intuitions come from? Can collocational knowledge be learned from exposure? Is simple co-occurrence frequency enough to learn them?

Engineering applications:

- collocations are different for different text types: discover them automatically to create dictionaries;
- translation systems have to replace a collocation in the source language with a valid collocation in the target language.

Can we discover collocations in corpora (large collections of text)?
The Naive Approach

The simplest way of finding collocations is counting. If two words occur together a lot, they form a collocation:

- go to a corpus;
- look for two word combinations (bigrams);
- count their frequency;
- select most frequent combinations;
- assume these are collocations.

Pointwise Mutual Information

As the previous example shows, if two words co-occur a lot in a corpus, it does not mean that they are collocations;

- if we have a set of candidate collocations (e.g., all co-occurrences of tea), then we can use $\chi^2$ to filter them (see Informatics 1B);
- however, this doesn’t work so well for discovering collocations from scratch;
- instead: use pointwise mutual information;
- intuitively, MI tells us how informative the occurrence of one word is about the occurrence of another word;
- words that are highly informative about each other form a collocation.
### Pointwise Mutual Information

**Example**

Take an example from the table:

\[
I(x; y) = \log \frac{f(x, y)}{f(x)f(y)} = \log \frac{c(x, y)}{N} \frac{N}{c(x)c(y)}
\]

\[I(\text{unsalted}; \text{butter}) = \log \frac{20}{14307668 \cdot 20} = 15.19\]

This means: the amount of information we have about *unsalted* at position *i* increases by 15.19 bits if we are told that *butter* is at position *i + 1* (i.e., uncertainty is reduced by 15.19 bits).

### Source Codes

**Definition: Source Code**

A source code \(C\) for a random variable \(X\) is a mapping from \(x \in X\) to \(\{0, 1\}^*\). Let \(C(x)\) denote the code word for \(x\) and \(l(x)\) denote the length of \(C(x)\).

Here, \(\{0, 1\}^*\) is the set of all finite binary strings (we will only consider binary codes).

**Definition: Expected Length**

The expected length \(L(C)\) of a source code \(C(x)\) for a random variable with the probability distribution \(f(x)\) is:

\[L(C) = \sum_{x \in X} f(x)l(x)\]

**Example**

Let \(X\) be a random variable with the following distribution and code word assignment:

<table>
<thead>
<tr>
<th>(x)</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f(x))</td>
<td>(\frac{1}{2})</td>
<td>(\frac{1}{4})</td>
<td>(\frac{1}{8})</td>
<td>(\frac{1}{8})</td>
</tr>
<tr>
<td>(C(x))</td>
<td>0</td>
<td>10</td>
<td>110</td>
<td>111</td>
</tr>
</tbody>
</table>

The expected code length of \(X\) is:

\[L(C) = \sum_{x \in X} f(x)l(x) = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3 = 1.75\]

**Definition: Non-singular Code**

A code is called non-singular if every \(x \in X\) maps into a different string in \(\{0, 1\}^*\).

- If a code is non-singular, then we can transmit a value of \(X\) unambiguously.
- However, what happens if we want to transmit several values of \(X\) in a row?
- We could use a special symbol to separate the code words.
- However, this is not an efficient use of the special symbol; instead use *self-punctuating* codes (prefix codes).
**Definition: Extension**

The extension $C^*$ of a code $C$ is:

$$C^*(x_1x_2 \ldots x_n) = C(x_1)C(x_2) \ldots C(x_n)$$

where $C(x_1)C(x_2) \ldots C(x_n)$ indicates the concatenation of the corresponding code words.

**Definition: Uniquely Decodable**

A code is called uniquely decodable if its extension is non-singular.

- If the code is uniquely decodable, then for each string there is only one source string that produced it;
- However, we have to look at the whole string to do the decoding.

**Example**

The following table illustrates the different classes of codes:

<table>
<thead>
<tr>
<th></th>
<th>Singular</th>
<th>Non-singular, not uniq. decodable</th>
<th>Uniq. decodable, not instant.</th>
<th>Instant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>010</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>01</td>
<td>11</td>
<td>110</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>10</td>
<td>110</td>
<td>111</td>
</tr>
</tbody>
</table>

**Definition: Prefix Code**

A code is called a prefix code (instantaneous code) if no code word is a prefix of another code word.

We don’t have to wait for the whole string to be able to decode it; the end of a code word can be recognized instantaneously.

**Example**

The code in the previous example is a prefix code. Take the following sequence: 01011111010.

The first symbol, 0, tells us we have an $a$; the next two symbols 10, have to correspond to $b$; the next three symbols have to correspond to a $d$, etc. The decoded sequence is: $abdcb$.

**Summary**

- Collocations are sequences of words that occur together;
- simple co-occurrence frequency in a corpus is not enough to discover collocations
- instead, use the pointwise mutual information of two words;
- a code is uniquely decodable if there is only one possible source sequence for every code sequence;
- a code is instantaneous if each code word has a unique prefix.