5.5.2 – Anomalies in Text Datasets

In the **one-hot encoding** framework, text data is typically

- **sparse**;
- **high-dimensional**, and
- **non-negative**.

Word vector representations help with the first two of these, but at the cost of removing the third (and reduced interpretability).

The anomaly detection models that have been discussed previously can be extended to text data, but with subtle differences.

Text Processing

Proximity-based models require either a sound distance function or a sound similarity function.

The **term frequency-inverse document frequency** representation is commonly-used for text processing: the term frequency for each word (i.e. how often it shows up in a document) is **normalized** by the inverse document frequency (i.e. in how many documents it appears).

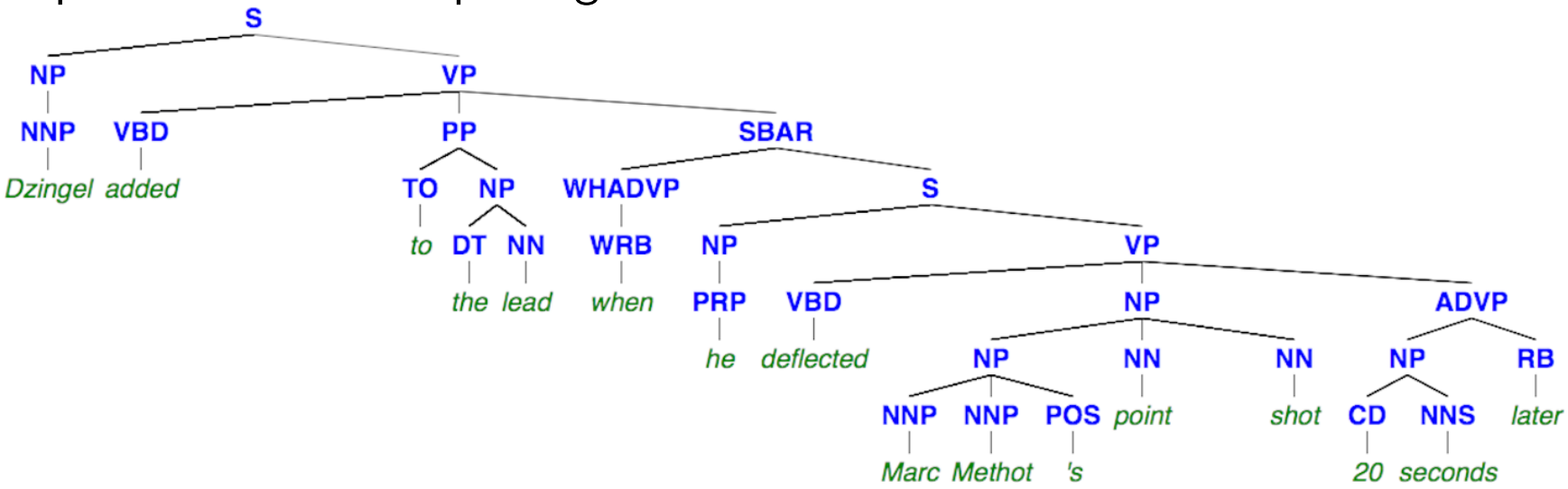
Semantic parsing is the process of converting a sentence in a natural language to a formal meaning representation.

The word **order** and its **type/role** provide the word's **attributes**.

Consider the sentence:

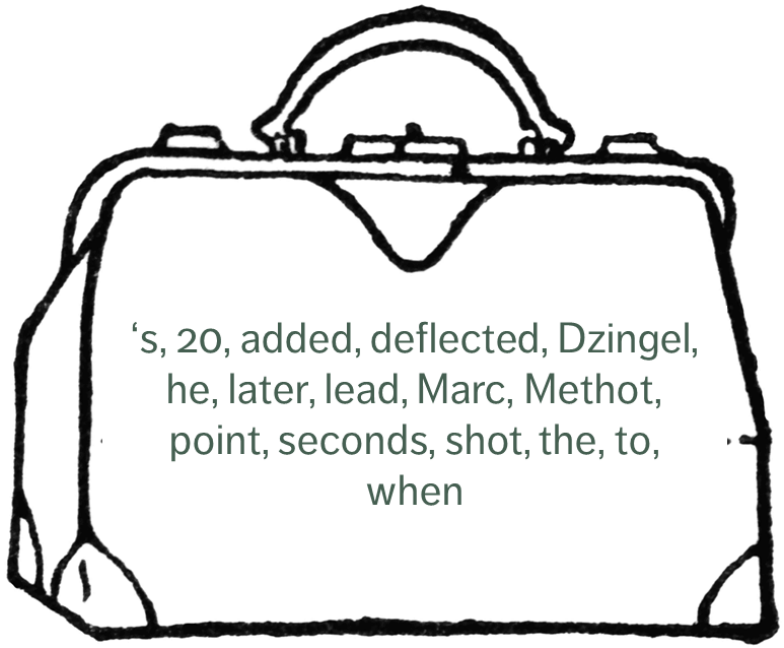
> "Dzingel added to the lead when he deflected Marc Methot's point shot 20 seconds later." (AP game recap, Ottawa Senators vs.Toronto Maple Leafs, 18-2-2017)

A potential semantic parsing tree is shown below:



By contrast, in the **Bag-of-Words** (BoW) framework, only the **presence** (or **absence**) of "words" (stems, $n-$grams, sentences, etc.) is important.

Relative frequencies provide information (intent, theme, feeling, etc.) about the corpus; the **words themselves** are attributes of the document.



In general, text data requires **extensive cleaning and processing**. But there are a number of challenges due to the nature of the data:

- what is an anomaly in the text?
- what is an outlier?
- are these concepts even definable?
- how do we deal with encoding errors?

⚠ Spelling mistakes and typographical errors are difficult to catch in large documents, even with spell-checkers; in the anomaly detection context, perhaps the text should not be cleaned up too thoroughly?

The process can be simplified to some extent with the help of **regular expressions** and **text pre-processing functions**.

Specific pre-processing steps vary depending on the problem:

- tweetish uses a different vocabulary than legalese
- same for a child who's learning to speak and a Ph.D. candidate defending her thesis

As is almost everything else related to text mining, the cleaning process is **strongly context-dependent**, and the order of pre-processing tasks can affect results.

Text Processing Options:

- convert all letters to **lower case** (avoid when seeking names)
- remove all **punctuation marks** (avoid if seeking emojis)
- remove all **numerals** (avoid when mining for quantities)
- remove all extraneous **white space**
- remove characters within **brackets** (avoid if seeking tags)
- replace **numerals with words**
- replace **abbreviations**

- replace **contractions** (avoid if seeking non-formal speech)
- replace all **symbols with words**
- remove **stop words** and **uninformative words** (language-, era- and context-dependent)
- **stem words** and **complete stems** to remove empty variations
  - "sleepiness", "sleeping", "sleeps", "slept" $\Longrightarrow$ "sleep"
  - the stem "operati" has different meanings in "operations research", "operating systems" and "operative dentistry"

Text Processing Challenges:

- **phonetic accent representation:** "ya new cah's wicked pissa!"
- **neologisms and portmanteaus:** "I'm planning prevenge"
- **poor translations/foreign words**, **puns and play-on-words**
- **mark-up, tags, and uninformative text:** `<b>`; `\verb`; ISBN blurb
- **specialized vocabulary:** "clopen"; "poset"; "retro-encabulator"
- **fictional names and places:** "Alderaan"; "Kilgore Trout"
- **slang and curses:** "skengfire"; `#$&#!`

Text must be stored to data structures with right properties:

- a **string** or **vector of characters**, with language-specific encoding;
- a **corpus** (collection) of text documents (with meta information)
- a **document-term matrix** (DTM) where the rows are documents, the columns are terms, and the entries are an appropriate text statistic (or the transposed term-document matrix (TDM))
- a **tidy text dataset** with one token (single word, $n-$gram, sentence, paragraph) per row

No magic recipe: best format depends on the problem at hand. But this step is **crucial**, both for semantic analysis and BoW.

	Document 1	Document 2	Document 3	⋯	Document N
Token 1	0	0	1	62	3
Token 2	0	1	0	61	2
Token 3	1	0	3	101	0
⋯	112	24	38	84	0
Token M	2	2	0	12	3
Sum	115	27	42	320	8

Sum
66
64
105
258
19

Document-Term Matrix

Consider a corpus $\mathcal{C} = \{d_1, \ldots, d_N\}$ consisting of N **documents** and M BoW **terms** $\mathcal{T} = \{t_1, \ldots, t_M\}$. Each document d contains M_d terms.

For instance, if

$$\mathcal{C} = \{\text{“the dogs who have been left out”}, \text{“who did that”}, \text{“my dogs breath smells like dog food”}\},$$

then $N = 3$, $M = 14$, and

$$\mathcal{T} = \{\text{“been”}, \text{“breath”}, \text{“did”}, \text{“dogs”}, \text{“food”}, \text{“have”}, \text{“let”}, \text{“like”}, \text{“my”}, \text{“out”}, \text{“smells”}, \text{“that”}, \text{“the”}, \text{“who”}\}.$$

The **relative term frequency** of term t in document d is

$$tf^*_{t,d} = \frac{\#\text{ of times term } t \text{ occurs in document } d}{M_d}.$$

$tf^*_{t,d}$		t													
		1 been	2 breath	3 did	4 dogs	5 food	6 have	7 let	8 like	9 my	10 out	11 smells	12 that	13 the	14 who
d	1	1/7	0	0	1/7	0	1/7	1/7	0	0	1/7	0	0	1/7	1/7
	2	0	0	1/3	0	0	0	0	0	0	0	0	1/3	0	1/3
	3	0	1/7	0	2/7	1/7	0	0	1/7	1/7	0	1/7	0	0	0

For instance, $1/3$ of the terms in document d_2 are "did;" $2/7$ of the terms in document d_3 are "dogs".

The **relative document frequency** of t is

$$df_t^* = \frac{\#\text{ of documents in which term } t \text{ occurs}}{N} = \frac{\sum_d \text{sign}(tf_{t,d}^*)}{N}.$$

In this example, all the terms occur in exactly one of the documents (not all the same), except for "dogs" and "who", which appear in two documents.

df_t^*	t													
	1 been	2 breath	3 did	4 dogs	5 food	6 have	7 let	8 like	9 my	10 out	11 smells	12 that	13 the	14 who
	1/3	1/3	1/3	2/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	2/3

None of the $df_{t,d}^*$ entries can be 0 (otherwise the corresponding document would be empty).

The **term frequency-inverse document frequency** of t in d is

$$tf\text{-}idf^*_t = -tf^*_{t,d} \times \ln(df^*_t).$$

$tf\text{-}idf^*_t$		t													
		1 been	2 breath	3 did	4 dogs	5 food	6 have	7 let	8 like	9 my	10 out	11 smells	12 that	13 the	14 who
d	1	0.16	0	0	0.06	0	0.16	0.16	0	0	0.16	0	0	0.16	0.06
	2	0	0	0.37	0	0	0	0	0	0	0	0	0.37	0	0.14
	3	0	0.16	0	0.12	0.16	0	0	0.16	0.16	0	0.16	0	0	0

Note that the entries for which $tf\text{-}idf^*_{t,d} = 0$ are also those for which the relative term frequency is $tf^*_{t,d} = 0$.

If all the documents contain the term t, then $df_t^* = 1$ and

$$tf\text{-}idf_{t,d}^* = -tf_{t,d}^* \times \ln(1) = 0,$$

and that specific term does not provide information.

If a term t rarely occurs in a document d, then $tf_{t,d}^* \approx 0$ and

$$tf\text{-}idf_{t,d}^* = 0 \times \ln(df_t^*) \approx 0.$$

Terms that appear relatively often only in a small subset of documents are crucial to understanding those documents **in the general context** of the corpus.

At the analysis stage, it is easy to forget where the data comes from and what it really applies to.

Text comes unstructured and unorganized. After processing, text is clean, but still unstructured. Bag of Words provides a framework for a structured numerical representation of text.

How does this affect the choice of text statistic in the DTM/TDM?

⚠ There is no sound mathematical justification to use the term frequency-inverse document frequency statistic. It may may not always be the ideal choice, but it is often used nonetheless.

Word vector representations change the nature of DTM.

Anomaly Detection Models

However the documents have been **normalized**, the corpus is now represented by a matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{D}_1 \\ \vdots \\ \mathbf{D}_N \end{bmatrix}.$$

For proximity-based methods, the similarity between documents is computed using the **cosine similarity**:

$$w(\mathbf{D}_i, \mathbf{D}_j) = \cos(\mathbf{D}_i, \mathbf{D}_j) = \frac{\mathbf{D}_i \cdot \mathbf{D}_j}{\|\mathbf{D}_i\| \|\mathbf{D}_j\|}.$$

Proximity-based algorithms can then be applied to this dataset, as they would to any numerical dataset. So can supervised algorithms, if data labels are available. The results, however, are not always ... satisfactory.