Session Four:

Alternative Text Data

CU SEAS IEOR 4723:

Financial Eng. for ESG Finance

4.1 *Alternative Text Data on ESG*

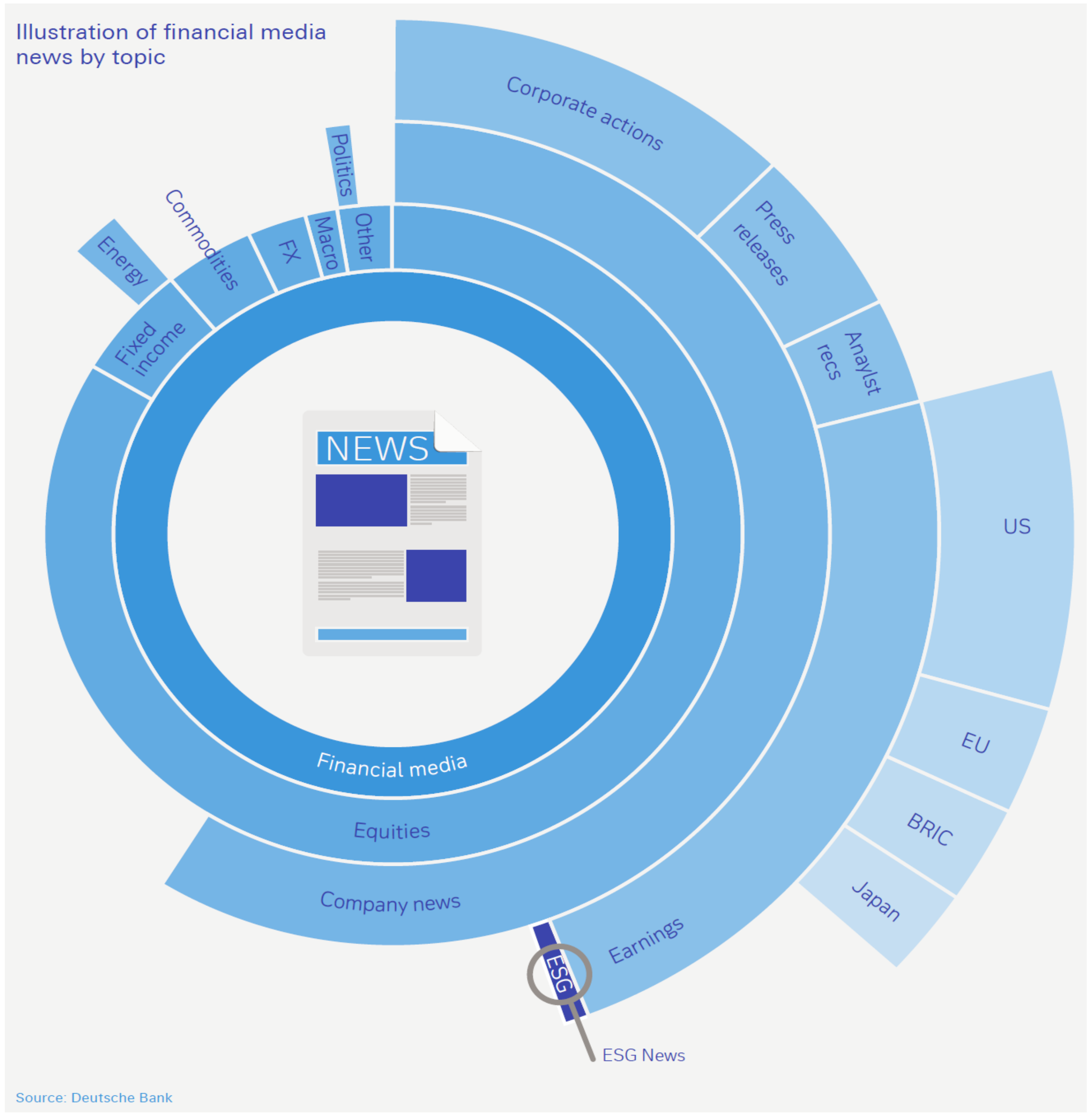
Majority of ESG Data Are Alternative…

Vast majority of ESG data are "alternative" i.e., does not come in standard tabular/numerical form

*Within the alternative category, most commonly used **(at least for now) is text data**, sourced from corporate reports as well as external media publications*

The amount of available text data is massive, which makes it impractical or even impossible to process it manually.

Information relevant to ESG is often dispersed among a wide corpus of texts. According to a 2018 sampling by Deutsche Bank, on average only 2% of the financial news were ESG-related (in 2018)



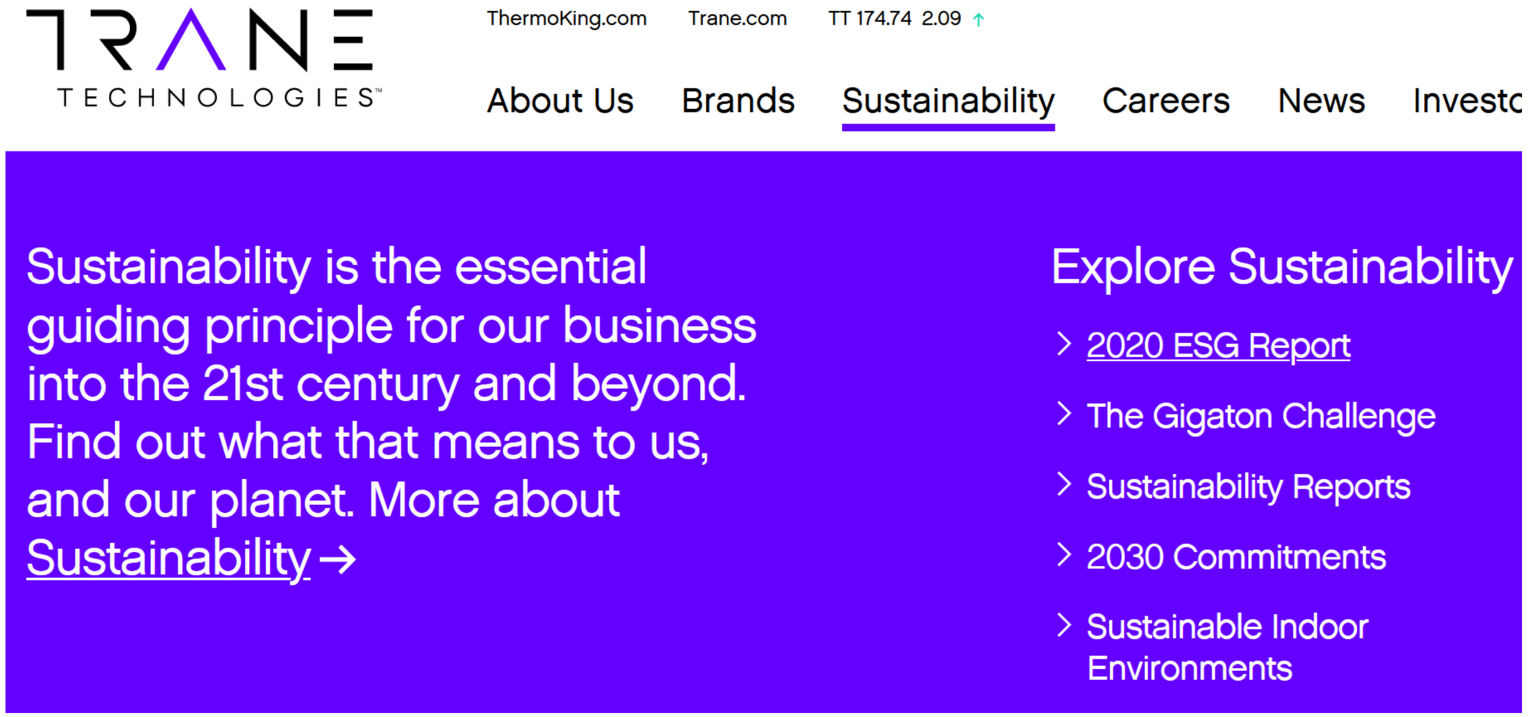
Source: Deutsche Bank

Corporate Reports on ESG

As noted earlier, much of the data requiring processing in the ESG Finance space comes in text form and may include Corporates' and Government's self-issued reports (accounting reports, as well as sustainability/ESG reports and press releases, etc.), news media articles about Corporates and Governments, social media postings, etc.

While corporate ESG disclosures remain voluntary at the moment, the proportion of firms choosing to self-report at least some aspects of their ESG information increases rapidly

In this session, we demonstrate a variety of applied text analyses of corporate ESG reports issued by a number of financial institutions



Sustainability is the essential guiding principle for our business into the 21st century and beyond. Find out what that means to us, and our planet. More about Sustainability →

Explore Sustainability

- 2020 ESG Report
- The Gigaton Challenge
- Sustainability Reports
- 2030 Commitments
- Sustainable Indoor Environments

Source: https://www.tranetechnologies.com/en/index/sustainability/sustainability-reports.html

4.2 *Natural Language Processing*

Natural Language Processing

We focus here on a brief introduction to Topic Modeling, as arguably the most critical area for applied ESG data analysis

For a more comprehensive introduction to NLP: “Getting started with natural language processing” by E. Kochmar, https://www.manning.com/books/getting-started-with-natural-language-processing or numerous other existing NLP textbooks

Python’s set of ML libraries ***scikit-learn*** provides a standard implementation of NLP models we apply here

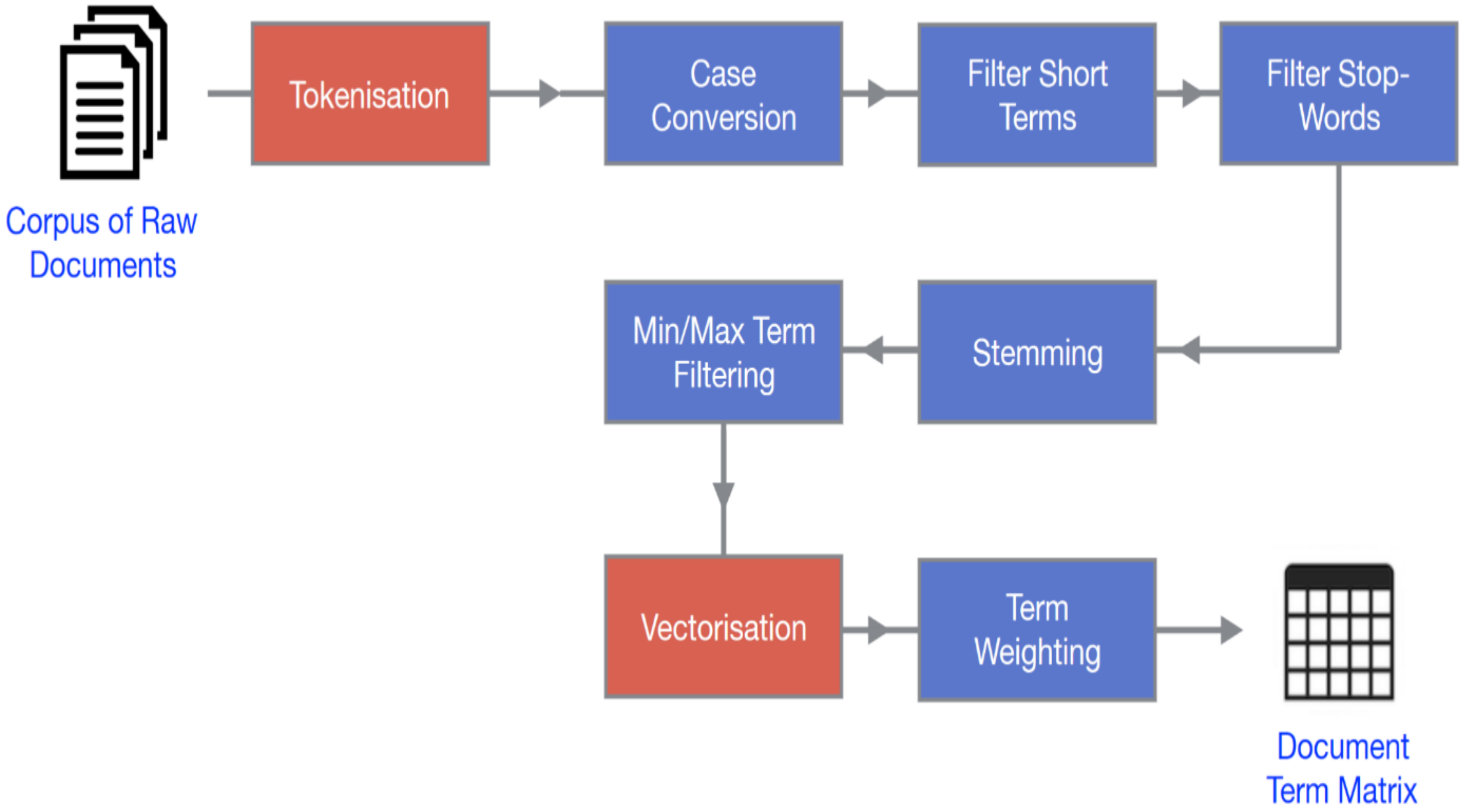
More cutting-edge NLP implementations also exist, especially those leveraging deep learning methodologies; most prominent recent example with wide practical applications is Bidirectional Encoder Representations from Transformers (BERT) by Google. An example of a BERT model trained on ESG reports: https://github.com/mukut03/ESG-BERT

Text Preparation

Typical text preprocessing steps summarized in a diagram

*Stemming/lemmatization is the one step not covered by **scikit-learn***

*Capabilities and certainly the inner workings of **spaCy** (which is capable of stemming) are out of scope; I do highly recommend the following post for a glance of its abilities though: https://explosion.ai/blog/parsing-english-in-python*



Source: http://derekgreene.com/

Text Preparation

Necessary preprocessing of text involves tokenization - splitting the unstructured texts into words separated by whitespace and punctuation - and “lemmatization” – converting words into their basic forms, removing plural suffixes, conjugation, etc. We focus here on analysis of texts in English; clearly, tokenization by whitespace would not work for many East Asian languages (or even German)

Other standard text preprocessing steps include:

- **Requiring minimum word length.** Very short words e.g. 1 or 2 letters long are excluded as likely not meaningful to the topic
- **Case conversion.** All texts are converted to lower case
- **Minimum and/or maximum document frequency.** Exclude words that appear in too few or too many documents in the corpus
- **Removing “stop words”**

Stop Words

Stop words are defined as words contained in a predefined list; these are common words that are assumed too common to be adding value to the topic. Words such as "have", "and", etc.

A generic list of English language stop words is a useful starting point, but it often helps to supplement the list with application-specific words as well

If we are comparing company-issued documents as is the case in many ESG applications, company names are added to the list of stop words: they appear often throughout the document and do not contribute to "topics"

In other contexts, company names should not be removed though: for example, if a particular company name (biggest client? Or biggest polluter?) appears across documents and may represent a "topic"

scikit-learn includes *sklearn.feature_extraction.text.ENGLISH_STOP_WORDS* preset list
(please note issues with this list raised in https://www.aclweb.org/anthology/W18-2502.pdf)

Exercises in this session borrow from and are inspired by a Databricks seminar
[Exercise A: Preprocessing of Corporate ESG Reports]

4.3 *Topic Modeling*

Topic Modeling

Topic Modeling is an unsupervised learning approach applied to text

Input: *is a corpus of unstructured text documents*
No explicit preexisting structure is required or accounted for

Output: *a list of:*
(a) topics and
(b) probabilities that a particular document is linked to a particular topic

(This is the probabilistic view we prefer here; an alternative definition of topic importance is possible, based on the properties of bag-of-words matrices)

Topic is defined by a few top-ranked words (see below how to rank words)

Bag-of-Words:
- Each document is represented by an n-dimensional vector, where n is the number of different words across all documents in the corpus
- Each cell in the vector represents the number of times the respective word appears in the document in question

This metric is referred to as “term frequency”
Class *sklearn.feature_extraction.text.CountVectorizer* implements the bag-of-words (“term frequency”) counting

Term Frequency – Inverse Document Frequency

Term frequency (TF) metric is straightforward in that TF is higher the more the particular word appears in the document. It does not account for whether the word itself is common (across a wider corpus)

To correct for this explicitly "term frequency – inverse document frequency" (TF-IDF) metric:

$$\text{TF-IDF} = \text{TF}(w, d) * (1 + \log_{10}(N / \text{DF}(w)))$$

where

- dash in "TF-IDF" is not a minus but part of the metric name
- TF(w, d) is term frequency of word w (number of times it appears) within document d
- DF(w) is "document frequency" of word w, i.e., number of documents in which w appears at least once among the total corpus of N documents

Example: let us say, the word "climate" appears 5 times in a given document d, but the word "water" appears 13 times (TF is 5 and 13 respectively).

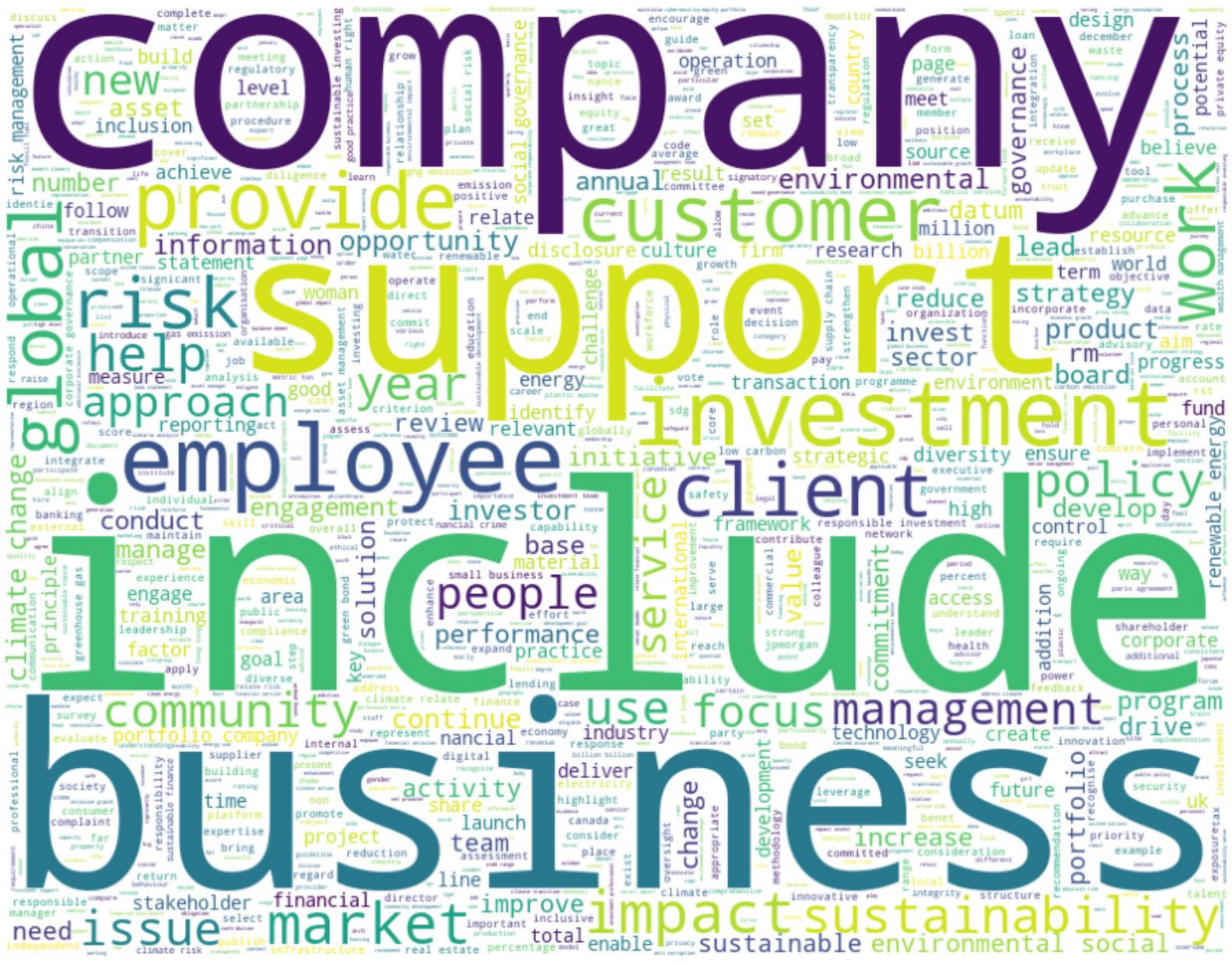
However, across a set of 10,000 documents, the word "water", being generally more common, appears in 5, 033 documents among 10,000; the word "climate" appears only in 100 of those. Then

TF-IDF("climate") = $5 * (1 + \log_{10}(10^4 / 10^2)) = 15$ while TF-IDF("water") ≈ 16.9

TF-IDF metric is also implemented in *sklearn.feature_extraction.text*; the class name is *TfidfVectorizer*

Term Occurrence in ESG Reports

Vectorizers' output can be displayed as beautiful greenish art



[Exercise B: Term Occurrence in ESG Reports]

Latent Dirichlet Allocation (LDA)

LDA is a probabilistic methodology of document classification, that is, topic assignment.
Non-probabilistic approaches to topic assignment exist and are also quite effective; we focus on the probabilistic approach here

LDA approach assumes that words in documents are multinomially distributed conditional on a small number of "topics" – unobservable variables, also multinomially distributed. The approach infers the joint topic distribution parameters from an observed document corpus sample

The basic algorithm:

- Choose the fixed total number of topics K
- Randomly make an initial assignment of one of K topics to each word in the document corpus
- For each topic t and each document d, compute P(t|d), a proportion of words in document d assigned topic t
- For each topic t and each word w, compute P(w|t), probability that assignment of topic t comes from word w (this is computed not directly but via Bayes rule)
- Reassign topics to each word w in each document to maximize P(t|d) * P(w|t)
- Repeat the last three steps a large number of times (until the assignments stabilize). The number of iterations is chosen a priori in practical implementations and does not rely on a convergence condition

LDA Continued

LDA is a probabilistic methodology of document classification, that is, topic assignment

The result of this algorithm is the inferred probability that document d belongs to topic t P(t|d), as well as most likely words under each topic, ranked by P(w|t)

We rely on our observation of several most likely words to interpret what each identified topic actually represents (its applied meaning, rather than its conceptual meaning, which is an unobservable multinomially-distributed random variable)

LDA is implemented by the *sklearn.decomposition.LatentDirichletAllocation* class

[Exercise C: Inference of Report Topics]

Outlier Topics

Some statements within each topic stand apart from the rest
We assume these statements describe actions/situations within each topic that are specific/unique to the company whose report the statement belongs to

Topic modeling exercise allowed us to classify all the statements across our corpus of report documents and to make distribution inferences regarding different topic represented

One question left unaddressed is specificity of topics to a particular company. A useful exercise that sheds some light on this topic specificity is detection of statement "anomalies". Knowing the inferred probabilities of belonging to a particular topic for each statement across our corpus, we group similar statements into clusters. The statements within each cluster that are relatively far away from the rest of the statements are these "anomalies"

Leverage one of the classical clustering methodologies for this (could have used other clustering techniques as well, but even this simple one appears fully adequate)

K-Means Clustering

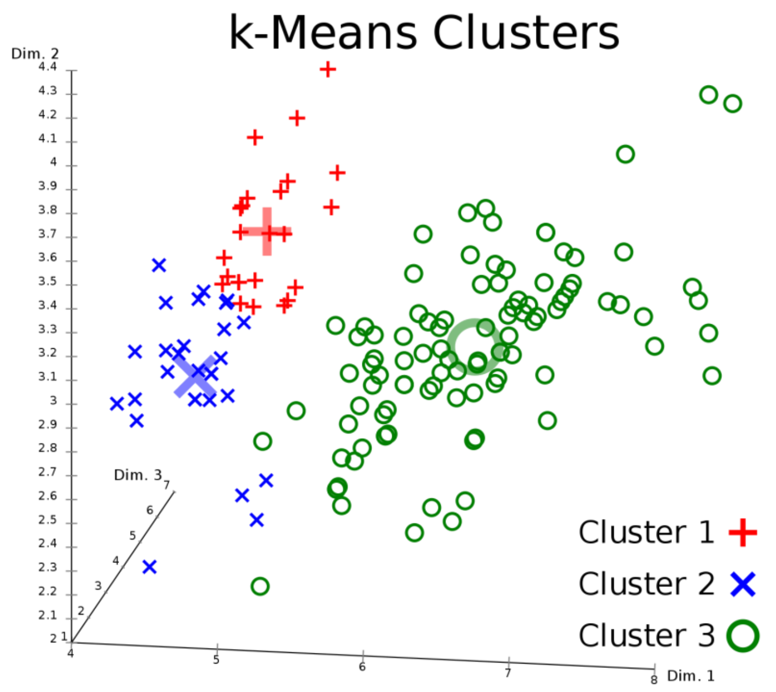
Clustering is a large category of unsupervised machine learning problems, not specific to NLP
There are numerous classical clustering algorithms of general use, as well as suited to specific applications
K-Means clustering algorithm is perhaps the simplest but adequate for the task

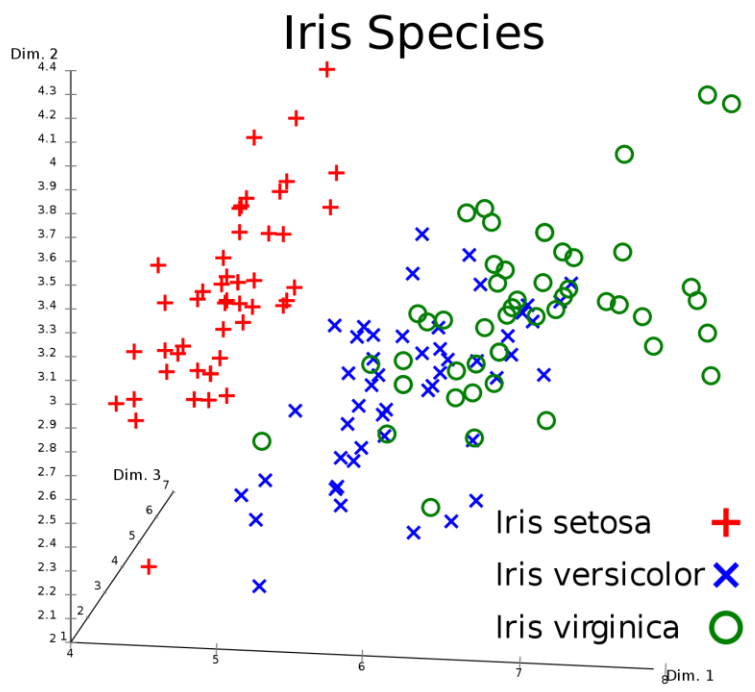
K-means clustering solves the problem of categorizing a number of points in n-dimensional space into "clusters" by distance, defined by a metric function

Any function of two vectors would do, provided it is nonnegative, symmetric, and satisfies the triangular inequality:

$$d(x, y) \leq d(x, z) + d(z, y).$$

We don't have to limit ourselves to Euclidean distance, but we need the ability to compute "average" points (centers of mass of a cluster)





Source: Wikipedia

Naïve K-Means Algorithm

The algorithm below:

- Choose the fixed total number of clusters *K*
- Make an initial assignment of cluster “centroid” points $m_1, \ldots, m_K$ (these points do not have to be a part of the data set)
- Repeat the following two steps many times until cluster choices stabilize (a typical choice of convergence condition is: centroids move by no more than a predefined amount between two last iterations):
  - Assign each point in the dataset to the nearest centroid (centroid *i* with the lowest value of $d(x, m_i)$). Data points assigned to the same centroid together constitute (a current iteration of) a cluster
  - Redefine new centroids to equal the averages (centers of mass) for each cluster:

$$m_i = \frac{1}{|\mathrm{M}_i|} \sum_{j \in \mathrm{M}_i} x_j$$

More efficient variants exist; also, more efficient initial choice of centroids can improve performance considerably

In ***scikit-learn***, K-means clustering is implemented as class *sklearn.cluster.KMeans*

[Exercise D: Identification of Outlier Topics]

Session IV Homework

HW

We have extracted the most common bigrams (by TF-IDF) in the ESG Reports corpus

- ***What are the three most common bigrams by TF alone?***
- ***What are the three most common bigrams (by TF-IDF) among the reports by (only) large banks?***