Pythonによるテキストマイニング

概要編

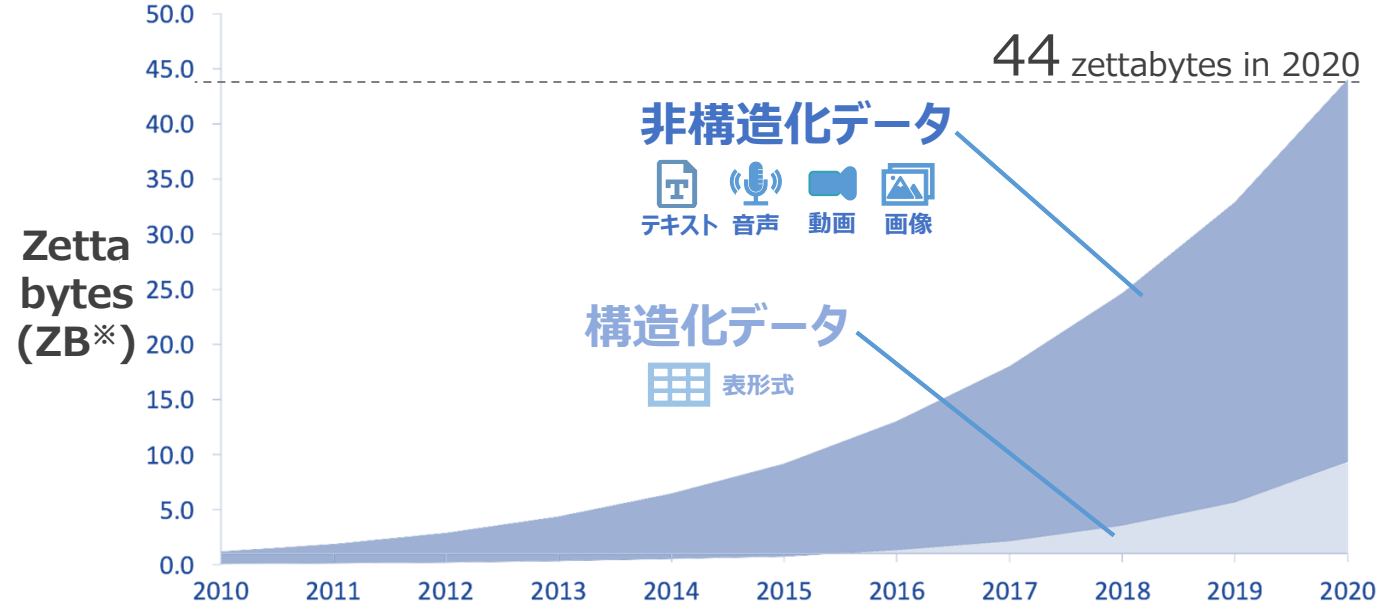
データサイエンス研究所

再掲

超ビッグデータ社会の加速とデータ活用ニーズ

■ビッグデータ社会の加速は止まらず、あらゆる企業の競争力に影響を与えている
■特に、製造業におけるデータ活用ニーズはひときわ高い

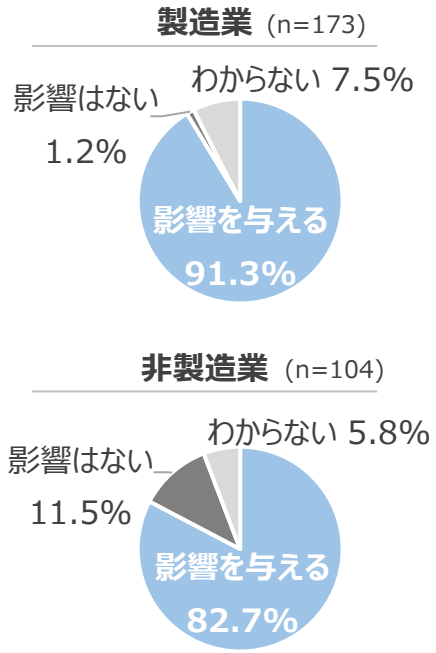
Capacity Growth by Data Type



※1ZB = 1兆GB = DVD2500億枚分

Source: IDC Market Spotlight (2016)
"Adoption of Object-Based Storage for Hyperscale Deployments Continues"

IoTやビッグデータの利活用が競争力に影響を与えるか？



Source: 日本経済団体連合会
「日本の国際競争力調査結果」(2016)

再掲

様々なデータの種類と活用例

■これまで扱ってきた数値データ以外にも、世の中には様々な種類のデータが存在しており、
今後、非構造化データの活用はますます加速すると考えられる

	データの種類	データの例	活用例	
構造化データ	クロスセクションデータ(断面データ)	▪売上データ ▪品質管理データ	▪売上予測、購買予測など多数	
構造化データ	時系列データ	▪株価データ ▪患者のバイタルデータ	▪株価予測 ▪容態急変のアラート	ここまでの時間で扱ってきたデータ
非構造化データ	画像	▪顔画像 ▪CT画像データ	▪顔認識 ▪病変の検出	
非構造化データ	動画	▪防犯カメラ	▪不審者の監視・検出	
非構造化データ	音声	▪人の会話 ▪機械の稼働音	▪音声入力・音声による機械との対話 ▪機械の作動異常の検出	
非構造化データ	テキスト	▪SNSのつぶやき ▪ニュース記事	▪トレンドワードの抽出 ▪ニュース記事の自動分類	本コンテンツで紹介

自然言語処理とテキストマイニング

■テキストデータは、他のデータとは異なり、そのままでは分析できないため、
「**自然言語処理**」（**NLP**）と呼ばれる前処理が必須となる

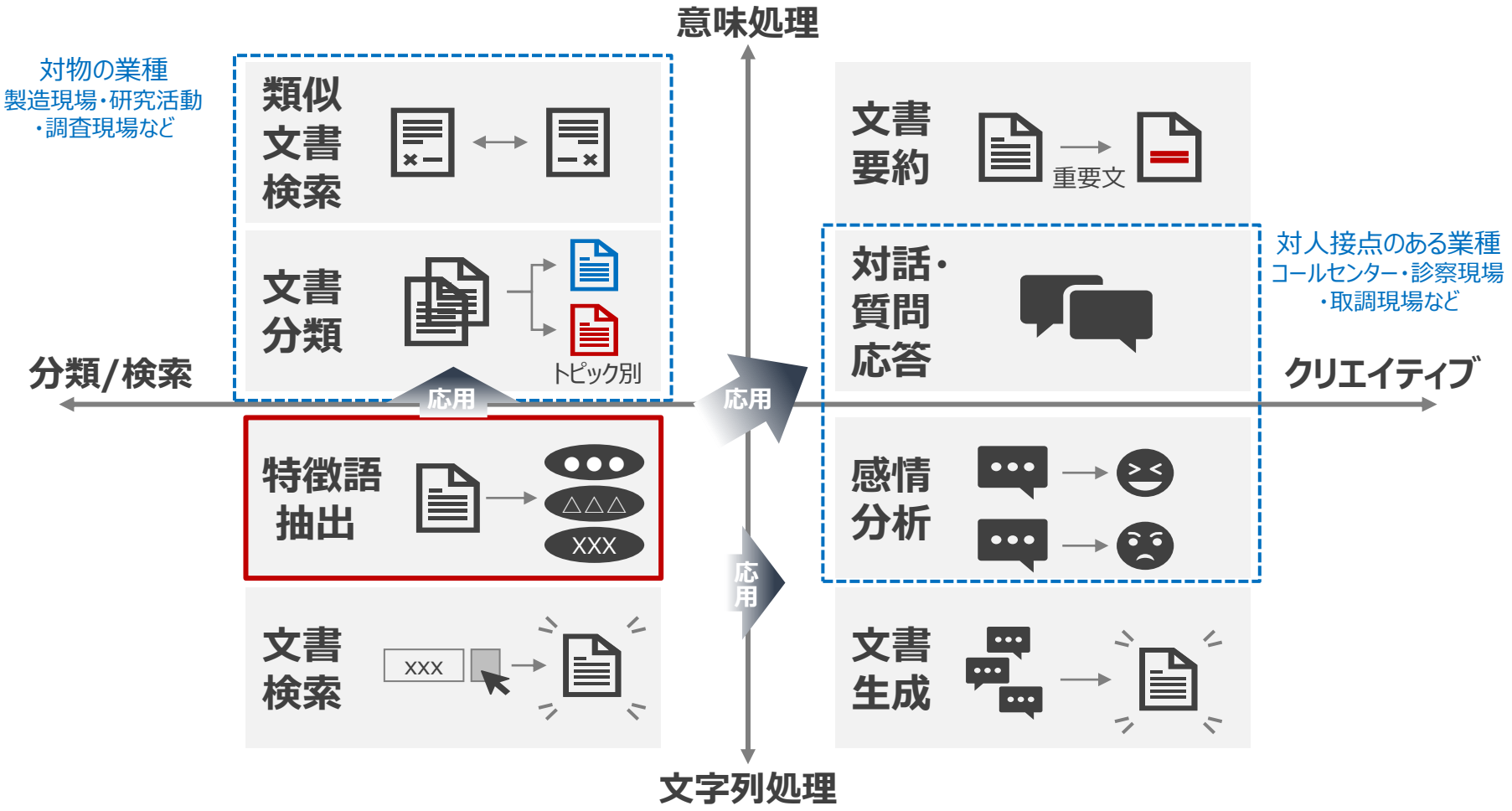
■特に、日本語は単語境界が曖昧なため、**形態素解析による単語分割**は必須の前処理である

テキストマイニング　≒　自然言語処理（**テキスト**前処理）　＋　データ**マイニング**

例：「鈴木さんは講義を受講した。」を自然言語処理すると・・・

テキストデータ活用の領域

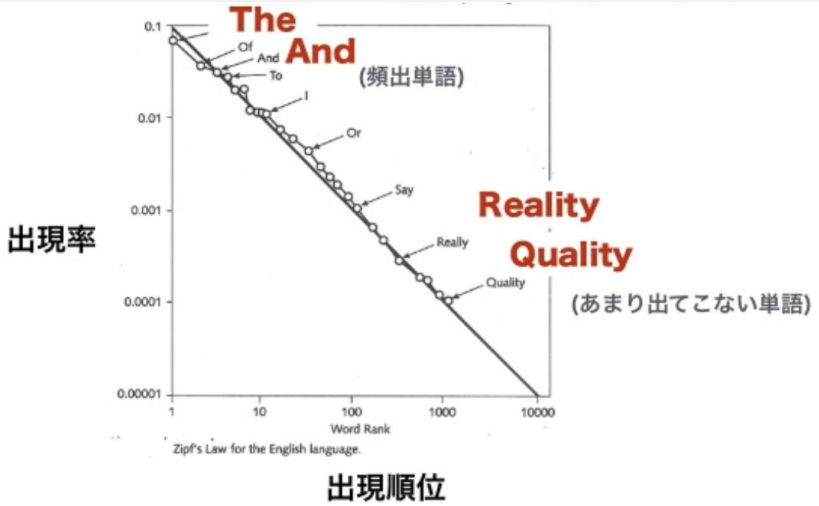
■近年は、人間の言語理解・発話力を代替するようなより高度なアプローチにシフトしつつあるが、依然として、「**特徴語の抽出**」は依然としてあらゆるアプローチの要素技術となりうる



出現単語の頻度集計とジップの法則

■テキストマイニングの最も基本的なアプローチは、形態素解析後に「出現単語の頻度」を集計

■一般に、出現単語のランキングは、ロングテール（上位の単語がほとんどを全体の占める状況）の構造を持ち、**ジップの法則**という経験則が成り立つことが知られている



出典：https://www.slideshare.net/mitsukioga/scala-38854156

ジップの法則
(Zipf's law)

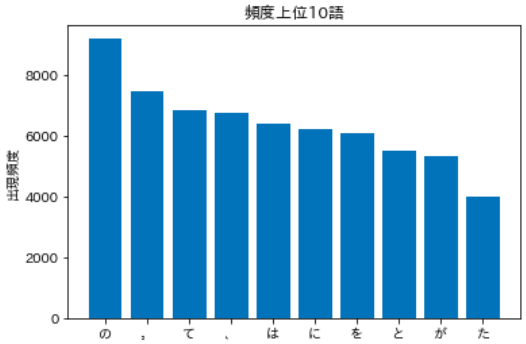
ある単語が全体に占める割合

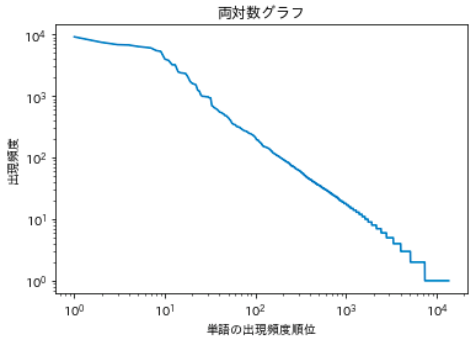
$$= \frac{10}{\text{その単語の出現順位}}$$

（つまり、両対数グラフで見ると直線になる）

日本語でも同様の傾向

（夏目漱石の「吾輩は猫である」を分析した人の事例）





出典：http://www.lovemysoulsoul.com/entry/2019/09/03/021807

単語の重要度を測る指標：TF-IDF

■一般に、**出現回数の多い単語**は「重要な単語」とみなせると考えられるが、単純に出現頻度で集計をとると、**一般語**や文書横断的なトピックに関する単語が上位にきやすい

■**TF-IDF**は、そのような「**ノイズ**」**を減らして本質的に重要な単語を見つける**ための指標である

「国内政治ニュース」の単語抽出例

記事1

安倍首相はx月x日・・・
トランプ大統領と会談し、
・・・
トランプ大統領の発言を受けて、**安倍首相**は、

記事2

安倍首相は**桜**を見る会の問題で、・・・
・・・
桜を見る会の参加者のうち、・・・

…

頻度集計　　頻度集計

#	単語	件数
1	**安倍首相**	10件
2	確認	8件
⋮	⋮	⋮
10	**トランプ大統領**	5件
⋮	⋮	⋮

#	単語	件数
1	**安倍首相**	7件
2	調査	5件
⋮	⋮	⋮
10	**桜**	4件
⋮	⋮	⋮

…

「よく登場する単語」が**重要とは限らない**
（政治ニュースに安倍首相が登場するのは当たり前！）

TF-IDFの定義

TF：単語の出現頻度
Term Frequency

$$= \frac{\text{ある文書内での}\textbf{単語Xの出現回数}}{\text{ある文書内での}\textbf{全単語の出現回数の和}}$$

IDF：逆文書出現頻度（単語のレア度）
Inverse Document Frequency

$$= \log \frac{\textbf{全文書数}}{\textbf{単語Xを含む文書の数}} + 1$$

※右辺に1を足すのは、IDF が 0 になるのを防ぐため

ある文書特異的な頻出単語を見つけるための指標：

TF-IDF = TF x IDF

テキストデータの「数値化」- Word2Vec

■画像データとは異なり、テキストはそのままでは数値データに変換して扱えない

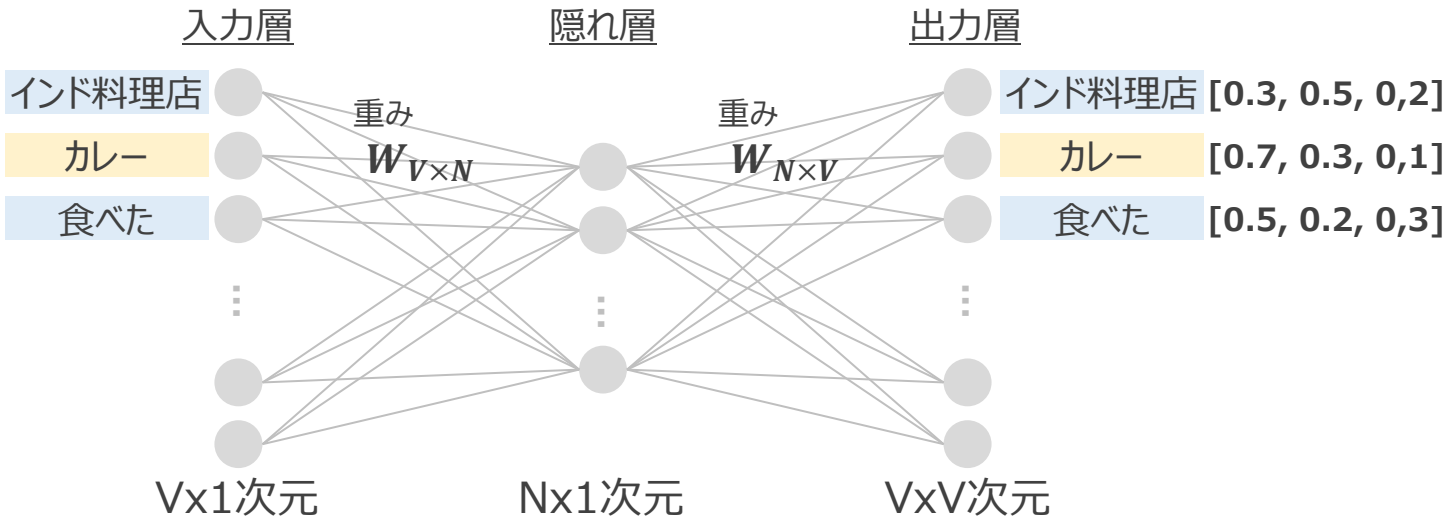
■そこで、ニューラルネットワークを用いて、前後の出現単語の「共起性」から、単語の類似度を測り、単語そのものを数値化（**ベクトル化**）する手法 **Word2Vec** が近年注目されている

私は昨日インド料理店でカレーを食べた

形態素解析

私 / 昨日 / インド料理店 / カレー / 食べた

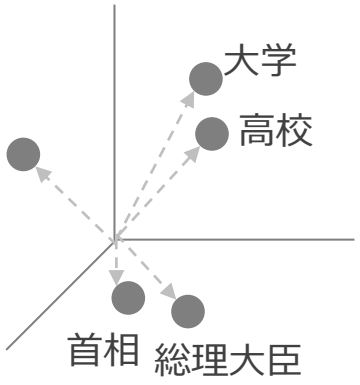
ベクトル化（数値化）



演習

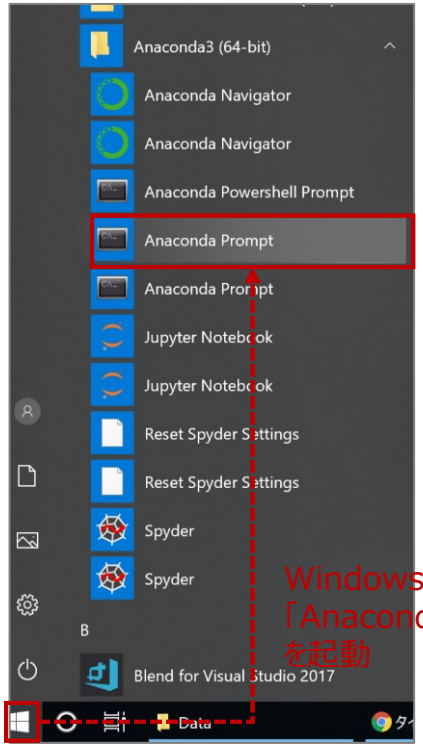
テキストベクトル化（Word2Vec）の体験

演習内容	Wikipediaデータを形態素解析し、 Word2Vecによりベクトル化した単語ベクトルモデルがある このモデルファイルを読み込み、類義語や単語同士の演算などを体験せよ
データ	Word2Vecモデル(Wiki).zip
ツール	Python (Jupyter Notebook)
時間	15分



テキストベクトル化（Word2Vec）の体験 (1/2)

▼自然言語処理パッケージ gensim のインストール（Anaconda Prompt 経由）



インストール画面が終われば
ライブラリのインストール完了

▼Word2Vecモデル(Wiki).zipを解凍して、中身3ファイルをJupyter Notebookファイルと同じ場所に配置

テキストベクトル化（Word2Vec）の体験 (2/2)

3.2 ライブラリの読み込み

```
from gensim.models import word2vec
```

3.3 Word2Vecモデル (Wikipediaから作成) の読み込み

```
model = word2vec.Word2Vec.load('wiki.model')

### modelファイルをこのjupyter notebookプログラム（.ipynb）と同じフォルダ内に配置する
# wiki.model
# wiki.model.trainables.syn1neg.npy
# wiki.model.wv.vectors.npy
```

3.4 単語ベクトルを用いた計算

3.4.1 類似ベクトルのランキング

```
model.wv.most_similar(positive = ['日本'], topn=20)
```

3.4.2 単語同士の引き算

```
model.wv.most_similar(positive=['女', '国王'], negative=['男'], topn=20)  #女＋国王-男

# positive=['足し算したい単語']
# negative=['引き算したい単語']
# topn=表示したい単語数
```

3.4.3 単語間の類似度算出

```
model.wv.similarity('大学', '高校')
```

End of File