@関西学院大学神学部
2021.01.18

講演会／ワークショップ
「テキストを対象とした認知人類学・宗教学研究」

Rを用いたテキスト分析実習

②RとMeCabを用いた日本語テキスト解析実習

明治大学　研究・知財戦略機構
佐藤浩輔

アウトライン（後半）

- 入力・クリーニングと前処理
  - 前処理の流れ
  - クリーニングの実習
- 分析Demo①『讃美歌21』のテキスト分析
  - 単語の頻度
  - キーワード抽出
  - 共起関係
- 分析Demo②CCLI TOP 100のテキスト分析
  - 単語の頻度
  - 出現頻度の比較

入力・クリーニングと前処理

入力・クリーニングと前処理

データの入力①

- まずデータを機械（コンピュータ）が扱えるようにする必要がある
- 自分で電子化する場合は（入力ミス/読取ミスがありうるので）特に念入りにクリーニングする

資料

電子化

1. Der Froschkönig oder der eiserne Heinrich

In den alten Zeiten, …

テキストデータ

入力・クリーニングと前処理

データの入力②

- テキストをデータ化しただけで十分か？
  - 十分でない
  - 分析に使う単位に分割・整理する必要がある

1. Der Froschkönig oder der eiserne Heinrich

In den alten Zeiten, …
…

2. Katze und Maus in Gesellschaft
…

テキストデータ

分割・整理

KHM	title	text
1	Der Froschkonig oder der eiserne Heinrich	In den alten Zeiten, …
2	Katze und Maus in Gesellschaft	Eine Katze hatte Bekanntschaft mit einer Maus, …
3	Marienkind	Vor einem großen Walde lebte ein Holzhacker mit seiner Frau, …
4	Marchen von einem, der auszog das Furchten zu lernen	Ein Vater hatte zwei Söhne, …
5	Der Wolf und die sieben jungen Geislein	Es war einmal eine alte Geiß,
…	…	…

表形式のデータ

入力・クリーニングと前処理

データの入力③

- ファイルフォーマット
  - 区切り文字を使って二次元の表形式のデータを格納
    - csv(comma-separated values; コンマ区切り)
    - tsv(tab-separated values; タブ区切り)
  - データを読み込む際には区切り文字に注意する
- メタデータ
  - メタデータ(metadata)：データに関する情報
  - 書誌情報
    - 著者、タイトル、発表年月日、etc.
  - 別ファイルに分けて後で結合してもよい
    - 結合のキーとなる変数(e.g. ID)がなければつける

入力・クリーニングと前処理

生のテキスト（平テキスト）

吾輩は猫である。名前はまだ無い。

どこで生れたかとんと見当がつかぬ。何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している。吾輩はここで始めて人間というものを見た。しかもあとで聞くとそれは書生という人間中で一番獰悪な種族であったそうだ。

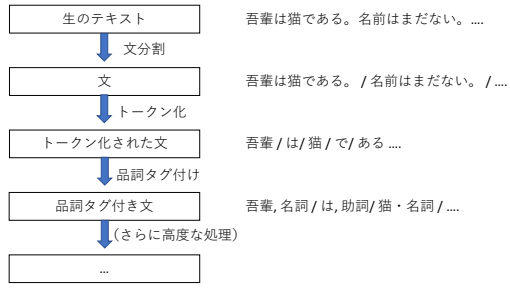
…

――夏目漱石『吾輩は猫である』

- 何のタグもつけられていないテキスト
- このままでは分析できない

入力・クリーニングと前処理

情報抽出アーキテクチャの例



Bird, Klein, Loper(2010) を改変

入力・クリーニングと前処理

文に分割

吾輩は猫である。/ 名前はまだ無い。/ どこで生れたかとんと見当がつかぬ。 / 何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している。 / 吾輩はここで始めて人間というものを見た。/しかもあとで聞くとそれは書生という人間中で一番獰悪な種族であったそうだ。 /

…

- 文ごとに分割されたテキスト

入力・クリーニングと前処理

単語(トークン)に分割

吾輩/は/猫/で/ある/。
名前/は/まだ/無い/。
どこ/で/生れ/た/か/とんと/見当/が/つか/ぬ/。
…

- 文を単語ごとに分割
  →こうやって分析できる単位に分割していく

入力・クリーニングと前処理

データ前処理

- クリーニング
  - ノイズを取り除く
  - 辞書を整備する
- 前処理
  - テキストの分割
    - 文書→文→単語に分割
  - 単語の処理
  - 様々なタグをつけたりする
- データハンドリング・構造化
  - 様々な分析手法が可能なようにデータを整える

クリーニングと前処理

クリーニング

- 分析にかけられるよう、データを整備する
  - 誤字脱字のチェック
  - 辞書の整備
    - 専門用語
    - 新語・死語
    - 方言
- ノイズが多く混じっていると結果が歪む

クリーニングと前処理

形態素解析

- 英語などの言語：
  - 単語と単語の間に切れ目がある
    →スペースで分割できる
- 日本語のような言語
  - 単語と単語との間に切れ目がない
  - 分割できるように「分かち書き」する必要がある
    →「形態素解析」という技術を使う

  - 形態素解析ソフトを使うことで、
    - 分かち書き
    - 品詞タグ付け

    を、一定の精度でまとめて処理することができる

クリーニングと前処理

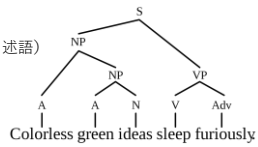
単語の処理

- ストップワード stop words
  - 話題の種類と関連を持たない語(e.g. a, 冠詞、助詞)
  - 分析に必要なければ除外する
- ステミング stemming
  - 派生語などを含めて同じ素性とみなす処理
    - e.g. operate→oper, operational→oper
  - 単語の形状をみて一律に処理する
    - e.g. Porter's stemmer
- 見出し語化 lemmatization
  - 単語を基本形に戻す処理
  - 構文情報を考慮して処理

クリーニングと前処理

タグ付け

- 付加情報をつける作業
  - 品詞 part of speech (POS)
    - 名詞
    - 動詞
    - 形容詞
    - 副詞
    - 助詞 etc.
  - 構文情報
    - 文法的な構造（e.g. 主語, 述語）



クリーニングと前処理

構造化

- 構造化の例
  - bag of words
    - 単語の頻度
    - 語順の情報は無視される
  - n-gram
    - 連続するn語の組み合わせ
      - 連続する２語→bigram
      - 連続する３語→trigram
  - 共起
    - 同じ文内に出現した語の組み合わせ

e.g. "This is a pen."

bag of words

this	1
is	1
a	1
pen	1

bigram

this-is	1
is-a	1
a-pen	1

共起

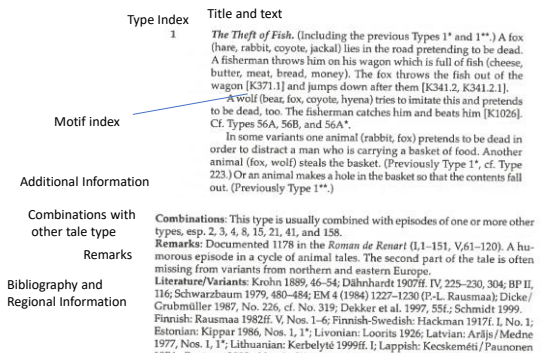
(this, is)	1
(this, a)	1
(this, pen)	1
(is, a)	1
(is, pen)	1
(a, pen)	1

→構造化されたデータを分析・可視化する

動物民話の分析(Nakawake & Sato, 2019)の場合

- コーパス
  - Aarne–Thompson–Uther type index(**ATU**)
    - Uther(2004) *The Types of International Folktales: A Classification and Bibliography*
  - 書籍を電子化(OCR)
- 対象
  - 「動物昔話」に分類される類型(ATU 1- ATU 299)
  - 類話を含む462項目のうち、話の内容について記述のある382項目
- 項目内容
  - 本文
  - メタ情報
    - Tale Type: ATUの中分類
    - Thompson Motif Index(**TMI**): 話の中に登場したモチーフ
    - 地域

本文の構造



1 ***The Theft of Fish.*** (Including the previous Types 1* and 1**.) A fox (hare, rabbit, coyote, jackal) lies in the road pretending to be dead. A fisherman throws him on his wagon which is full of fish (cheese, butter, meat, bread, money). The fox throws the fish out of the wagon [K371.1] and jumps down after them [K341.2, K341.2.1].

A wolf (bear, fox, coyote, hyena) tries to imitate this and pretends to be dead, too. The fisherman catches him and beats him [K1026]. Cf. Types 56A, 56B, and 56A*.

In some variants one animal (rabbit, fox) pretends to be dead in order to distract a man who is carrying a basket of food. Another animal (fox, wolf) steals the basket. (Previously Type 1*, cf. Type 223.) Or an animal makes a hole in the basket so that the contents fall out. (Previously Type 1**.)

Combinations: This type is usually combined with episodes of one or more other types, esp. 2, 3, 4, 8, 15, 21, 41, and 158.

Remarks: Documented 1178 in the *Roman de Renart* (I,1–151, V,61–120). A humorous episode in a cycle of animal tales. The second part of the tale is often missing from variants from northern and eastern Europe.

Literature/Variants: Krohn 1889, 46–54; Dähnhardt 1907ff. IV, 225–230, 304; BP II, 116; Schwarzbaum 1979, 480–484; EM 4 (1984) 1227–1230 (P.-L. Rausmaa); Dicke/Grubmüller 1987, No. 226, cf. No. 319; Dekker et al. 1997, 55f.; Schmidt 1999. Finnish: Rausmaa 1982ff. V, Nos. 1–6; Finnish-Swedish: Hackman 1917f. I, No. 1; Estonian: Kippar 1986, Nos. 1, 1*; Livonian: Loorits 1926; Latvian: Arāljs/Medne 1977, Nos. 1, 1*; Lithuanian: Kerbelytė 1999ff. I; Lappish: Kecskeméti/Paunonen

入力・クリーニング

1 ***The Theft of Fish.*** (Including the previous Types 1* and 1**.) A fox (hare, rabbit, coyote, jackal) lies in the road pretending to be dead. A fisherman throws him on his wagon which is full of fish (cheese, butter, meat, bread, money). The fox throws the fish out of the wagon [K371.1] and jumps down after them [K341.2, K341.2.1].

A wolf (bear, fox, coyote, hyena) tries to imitate this and pretends to be dead, too. The fisherman catches him and beats him [K1026]. Cf. Types 56A, 56B, and 56A*.

In some variants one animal (rabbit, fox) pretends to be dead in order to distract a man who is carrying a basket of food. Another animal (fox, wolf) steals the basket. (Previously Type 1*, cf. Type 223.) Or an animal makes a hole in the basket so that the contents fall out. (Previously Type 1**.)

OCRで読み込み

1. The Theft of Fish. (Including the previous Types 1* and 1**.) A fox (hare, rabbit, coyote, jackal) lies in the road pretending to be dead. A fisherman throws him on his wagon which is full of fish (cheese, butter, meat, bread, money). The fox throws the fish out of the wagon [K371.1] and jumps down after them [K341.2, K341.2.1].

A wolf (bear, fox, coyote, hyena) tries to imitate this and pretends to be dead, too. The fisherman catches him and beats him [K1026]. Cf. Types 56A, 56B, and 56A*.

In some variants one animal (rabbit, fox) pretends to be dead in order to distract a man who is carrying a basket of food. Another animal (fox, wolf) steals the basket. (Previously Type 1*, cf. Type 223.) Or an animal makes a hole in the basket so that the contents fall out. (Previously Type 1**.)

前処理

- 本文のみを抽出
  - 正規表現で関係のない部分を削除
    - モチーフのタグ
    - 参照情報
- PythonのNLTKモジュールを用い、タグ付け
  - 品詞ごとに抽出
  - 文章を見出し語化(lemmatize)
  - synsetの中に「動物」の意味を含む語を抽出
    →リスト化
  - リストを用いて、物語の中に登場する動物をインデックス化
  - 再度コーパスを参照して有効な件数を数える

The Theft of Fish. ~~(Including the previous Types 1 * and 1 **.)~~A fox (hare, rabbit, coyote, jackal) lies in the road pretending to be dead. A fisherman throws him on his wagon which is full of fish (cheese, butter, meat, bread, money). The fox throws the fish out of the wagon ~~[K371.1]~~ and jumps down after them ~~[K341.2, K341.2.1]~~. A wolf (bear, fox, coyote, hyena) tries to imitate this and pretends to be dead, too. The fisherman catches him and beats him ~~[K1026]~~. ~~Cf. Types 56A, 56B, and 56A *.~~ In some variants one animal (rabbit, fox) pretends to be dead in order to distract a man who is carrying a basket of food. Another animal (fox, wolf) steals the basket. ~~(Previously Type 1 *, cf. Type 223.)~~ Or an animal makes a hole in the basket so that the contents fall out. ~~(Previously Type 1 **.)~~

等位の名詞の処理

- ‘fox (hare, jackal)’ , ‘lion’
  - (fox, lion), (hare, lion), (jackal, lion)
  - ~~(fox, hare), (fox jackal) (hare, jackal)~~

→すべての組み合わせを先に求め、等位のものを除く

The Theft of Fish. A fox (hare, rabbit, coyote, jackal) lies in the road pretending to be dead. A fisherman throws him on his wagon which is full of fish (cheese, butter, meat, bread, money). The fox throws the fish out of the wagon and jumps down after them A wolf (bear, fox, coyote, hyena) tries to imitate this and pretends to be dead, too. The fisherman catches him and beats him. In some variants one animal (rabbit, fox) pretends to be dead in order to distract a man who is carrying a basket of food. Another animal (fox, wolf) steals the basket. Or an animal makes a hole in the basket so that the contents fall out.

['hyena', 'wolf' 'rabbit', 'coyote', 'jackal', 'bear', 'man', 'fox', 'fish'],

id	atu_id	animals
1	1	fish, fox, man, wolf, bear, jackal, hare, hyena, coyote, rabbit
4	2	fish, men, fox, wolf, bear, dog
5	2A	jackal, wolf, men, fox
6	2B	fish, wolf, fox
8	2D	bear, wolf, fox
9	3	bear, wolf, fox

id	#	motif
1	1	K371.1,K341.2,K341.2.1,K1026
4	2	K1021
5	2A	K1021.1,J758.1,J341.1
6	2B	K1021.2
9	3	K473,K522.1,cf.K1875
10	3*	K1022.3
11	4	K1241,K1818

上：動物の出現
右：抜き出したモチーフ

https://github.com/satocos135/animalfolktale-analysis

実習

事前準備

- ソフトウェア
  - R version 3.4.3以上
  - MeCab
    - Windowsの場合はShift-JIS版がインストールされていること
    - IPA辞書がインストールされていること
- Rパッケージ
  - RMeCab
  - tidyverse
    - dplyr(tidyverseに含まれる)
    - stringr(tidyverseに含まれる)

※2020/12/28のトークの内容を理解している前提で進めます

Rによる処理の基礎

RMeCabの使い方

- RMeCab
  - MeCab: 日本語の形態素解析エンジン
  - RMeCab: R上でMeCabを使うためのパッケージ(石田, 2017)
    - 作者のページに各関数の詳しい解説がある
      - http://rmecab.jp/wiki/index.php?RMeCabFunctions
    - RMeCabC(): 文字列を形態素解析して返す
    - docMatrixDF(): フォルダ内のファイルごとに解析する
    - docDF(): データフレームの行ごとに解析する
    - 解析時に辞書ファイルを指定できる

クリーニングの実際

- 例文：島根県での聞き取り調査(2019)の一コマ
  Q:「やまいり？やまはいって木を丸める？」
  A:「切ってきて、丸めて、家の屋根にしげておくん。それはてんかごめんけどどこのやまいってきってきてもよいということになっちょった。」
- これを形態素解析にかけてみる

1. 動詞: '切っ'
2. 助詞: 'て'
3. 動詞: 'き'
4. 助詞: 'て'
5. 記号: '、'
6. 動詞: '丸め'
7. 助詞: 'て'
8. 記号: '、'
9. 名詞: '家'
10. 助詞: 'の'
11. 名詞: '屋根'
12. 助詞: 'に'
13. 形容詞: 'しげ'
14. 助詞: 'て'
15. 動詞: 'おく'
16. 助動詞: 'ん'
17. 記号: '。'
18. 名詞: 'それ'
19. 助詞: 'は'

方言または誤字・聞き取りミス

20. 名詞: 'てんか'
21. 感動詞: 'ごめん'
22. 接続詞: 'けど'
23. 名詞: 'どこ'
24. 助詞: 'の'
25. 助詞: 'や'
26. 動詞: 'まいっ'
27. 助詞: 'て'
28. 動詞: 'きっ'
29. 助詞: 'て'
30. 動詞: 'き'
31. 助詞: 'て'
32. 助詞: 'も'
33. 形容詞: 'よい'
34. 助詞: 'という'
35. 名詞: 'こと'
36. 助詞: 'に'
37. 動詞: 'なっ'
38. 名詞: 'ちょ'
39. 助詞: 'っ'
40. 助動詞: 'た'
41. 記号: '。'

「てんかごめん」（天下御免）の誤認識

誤字・聞き取りミス？

「やま　いって」が「や　まいって」に

方言

クリーニングの実際

クリーニング方略

- ノイズを減らす
  - 誤字・脱字をチェックする
  - 句読点を入れる
  - 漢字に変換する
  - 表記揺れを減らす
- 固有名詞/専門用語/特殊な言い回しへの対応
  - 辞書を整備する
  - 標準語に変換する
  - 領域固有の知識(e.g. 方言)を持つ

クリーニングの実際

修正の例

「切ってきて、丸めて、家の屋根にしげておくん。それはてんかごめんけどどこのやまいってきってきてもよいということになっちょった。」

「切ってきて、丸めて、家の屋根にすげておくん。それは**天下御免**でどこの**山**に**行**って**切**ってきてもよいということになっていた。」

クリーニングの実際

★MeCabの辞書の設定

- 追加辞書用csvを作る
  - MeCab/dic/ipadic配下に辞書のcsvがたくさん入っているので参考にする
- コンパイルする
  - MeCab/bin配下のmecab-dict-index.exeを使う
- RMeCabの関数実行時に辞書ファイルを指定

クリーニングの実際

★辞書CSVの作成

指定不要　　IPA 品詞体系を参考にする

表層形	左文脈ID	右文脈ID	コスト	品詞	品詞細分類1	品詞細分類2	品詞細分類3	活用型	活用形	原形	読み	発音
けん	*	*	1000	助詞	接続助詞	*	*	*	*	けん	ケン	ケン

同じカテゴリの単語を参考にする

けん,*,*,1000,助詞,接続助詞,*,*,*,*,けん,ケン,ケン

クリーニングの実際

★辞書のコンパイル

MeCabフォルダに移動し、

```
¥bin¥mecab-dict-index.exe -d デフォルト辞書フォルダ
-u 出力ファイル名
-f 文字エンコーディング（入力ファイル）
-t 文字エンコーディング（出力ファイル）
入力ファイル名
```

を実行（行を分けずに入力する）

・実行例

```
¥bin¥mecab-dict-index.exe -d dic¥ipadic -u
c:¥Users¥satoc¥projects¥nlp2019¥example.dic -f shift-jis -t
shift-jis c:¥Users¥satoc¥projects¥nlp2019¥shimane.csv
```

```
MeCab('まあそういうことで、余分な話はいいけん。')
```

1. 副詞:'まあ'
2. 連体詞:'そういう'
3. 名詞:'こと'
4. 助動詞:'で'
5. 記号:'、'
6. 名詞:'余分'
7. 助動詞:'な'
8. 名詞:'話'
9. 助詞:'は'
10. 形容詞:'いい'
11. 助詞:'け'
12. 助詞:'ん'
13. 記号:'。'

```
MeCab('まあそういうことで、余分な話はいいけん。', dic='example.dic')
```

1. 副詞:'まあ'
2. 連体詞:'そういう'
3. 名詞:'こと'
4. 助動詞:'で'
5. 記号:'、'
6. 名詞:'余分'
7. 助動詞:'な'
8. 名詞:'話'
9. 助詞:'は'
10. 形容詞:'いい'
11. 助詞:'けん'
12. 記号:'。'

クリーニングの実際

どれくらいクリーニングすればよいか？

- テキストデータは一般に膨大
  - 人手で完璧にチェックすることは不可能
    - 自動で大量に処理できることのメリットが失われる
  - 事前に何が問題になるかはわかりづらい
    - 探索的なプロセス / データセットそれぞれの固有の問題
- 無難なアプローチ：分析しながら、漸進的に改善
  - ノイズになっている個所を見つけたら対応する
    - 固有名詞や方言 / 形態素解析の誤認識
  - 単語の頻度表をチェックする
    - 当然多くなるべき単語が多くなっているか
    - 不可解な言葉が多くなっていないか
  - 分析
    - いつでも元データから最新のデータを作れるようコードを保存する
    - コアとなる分析に関係する部分は入念にチェックする
    - 日ごろから様々なエラーの可能性を検討しておく

クリーニングの実際

クリーニングのヒント

- 辞書に載っている表記に変える
  - ひらがな・カタカナを漢字にする
  - 伸ばし棒なのか母音なのか
- 特殊な言い回しに対応する
  - MeCabの辞書に追加する
    - 追加する際の情報は標準語の対応する単語を参考にする
  - 辞書で対応できそうになければ、一括で置換する
    - 正規表現での検索・置換に対応したエディタ等を使う

分析DEMO
①『讃美歌21』のテキスト分析

『讃美歌21』の分析

- 『讃美歌21』
  - 日本基督教団讃美歌委員会によって、1997年編纂・出版された讃美歌集
    - 日本基督教団：1941年に日本国内のプロテスタント33教派が合同して設立した合同教会
  - 内容
    - 580の讃美歌を含む
      - 93番は礼拝文
      - ひとつの番号に複数の曲が含まれることがある
        - e.g. 39番「ハレルヤ」は40-1~40~7まである
    - 定旋律のものから20世紀の作曲のものも含む

歌詞の分析にあたって

- 歌詞：音楽に合わせて歌われるテキスト
  - 通常の散文ではない
    - 繰り返しのフレーズがある場合がある
      - 同じ単語の出現回数がかなり増える可能性がある
  - 音楽的構造を持つ
    - 形式：一般的なコラールはAAB形式
    - 音楽的要素・修辞との関係性も検討可能

分析①：『讃美歌21』

データセットの構造

番号	タイトル	節番号	歌詞
id	title	verse	lyric
001	主イェスよ、われらに	1	主イェスよ、われらに きよきみ顔向け、：聖霊を…
001	主イェスよ、われらに	2	礼拝につどえる み民をつよめて、：その口を開き、…
001	主イェスよ、われらに	3	われ、み顔をあおぎ、喜びに満ちて、：主をたたえ…
001	主イェスよ、われらに	4	父と子と聖霊 ひとりのみ神に、：ほまれとみ栄え…
002	聖なるみ神は	1	聖なるみ神は われらの集いに：いま共にいます。…
002	聖なるみ神は	2	救いのみ神は 悔いたる心に：愛をもてせまり、：罪、…
002	聖なるみ神は	3	いのちのみ神は 主の民守りて、：糧をあたえたもう。…
003	扉を開きて	1	扉を開きて われを導き、：まことの光と 慰め満つる…
003	扉を開きて	2	わが主よ、 みまえに われは来たりぬ。：われらの…
003	扉を開きて	3	おそれおののきて みまえに来たり、：こころもからだ…
003	扉を開きて	4	導く星なる 主のみことばを：われらに与えて つねに…
003	扉を開きて	5	語りたまえ、主よ、祈るわれらに。：いのちの…

ゼロパディングした文字列として入力

行区切りを全角コロンに

処理の方針

- 「**日本語**の**歌詞**の讃美歌を分析する」
  - 歌詞でないものを除く
    - 93番（礼拝文）→除外
    - 114番「民よ、主に仕えよ」→朗読部分を除く
    - 154番「みことばはわたしの喜び」→朗読部分を除く
    - 34番「キリエ、キリエ・エレイソン」→祈り部分を除く
  - 日本語を含まないものは歌詞の分析から除く
    - 33番「キリエ、キリエ」
    - 38番「グローリア、グローリア」
    - 39-1~39-7番「ハレルヤ」
    - 40-1~40-8番「アーメン」
    - 43-1~43-2番「マラナ・タ」
    - 176番, 177番「マニフィカート」
  - アルファベットで記述されている部分は除く

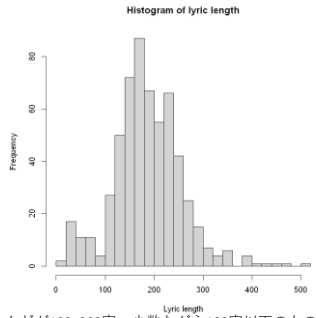
処理の方針

- 答唱/くりかえしなど
  - 一回分だけデータに含める
- 括弧内の読み仮名等は削除する

分析の方針

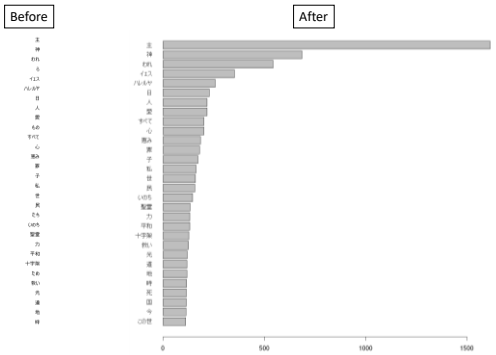
- 全体の特徴をつかむ
  - 全体の頻出語彙の抽出
  - 数量的特徴
- 讃美歌はテーマごとに配列されている
  - 各テーマの特徴を表すような語を抜き出せるか？

歌詞の長さの分布

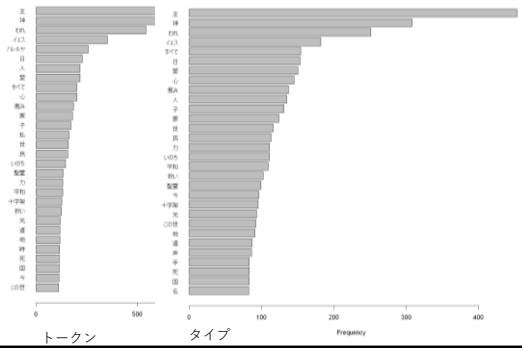


- ほとんどが100~300字。少数ながら100字以下のものもある。ごくわずかに400字を超えるものがある。

上位頻出後：ストップワードの除外



トークン vs. タイプ



トークン　タイプ

メタデータは役に立つ

- メタデータ：データに関するデータ
  - 事前に各文書の属性がわかっているならそれを使ったほうがよい場合が多い
    - ミックスジュースを分析するより一種類のジュースを分析するほうが楽

『讃美歌21』のメタデータ

id	title	major_id	major	minor_id	minor	key
001	主イエスよ、われらに	1	礼拝	1	招き	F major
002	聖なるみ神は	1	礼拝	1	招き	D major
003	扉を開きて	1	礼拝	1	招き	Bb major
004	世にあるかぎりの	1	礼拝	1	招き	G major

TF-IDF

- TF-IDF: **T**erm **F**requency – **I**nverse **D**ocument **F**requency
  - 文書群について、単語がどれくらい特徴的かを表す指標
    →文書のキーワードを抜き出すために使える
    - TF:Term Frequency 単語頻度
      - それぞれの文書について、その単語が出てくる程度
    - IDF:Inverse Document Frequency 逆文書頻度
      - 全体の文書のうち、その単語を含む文書の程度（の逆数）
      - 複数の文書に出現する単語ほど特徴的でない
    - TF-IDF
      - TFとIDFの積
      - 少数の文書に頻出する単語ほど強く重みづける
  - 経験的な指標：理論的な基礎ははっきりしないが、**有用**なためテキストマイニングや検索エンジンなどで幅広く使われている
  - 共通する単語は低く重みづけられるので、同系統の文書を分析するときには注意が必要

TF-IDFの計算方法

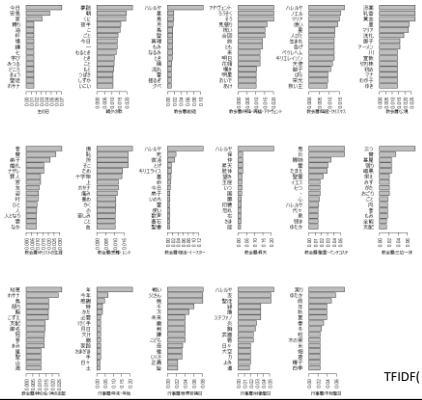
$$TF = \frac{\text{当該の単語の出現回数}}{\text{文書内の総単語数}}$$

$$IDF = \log \frac{\text{総文書数}}{\text{当該の単語を含む文書の数}}$$

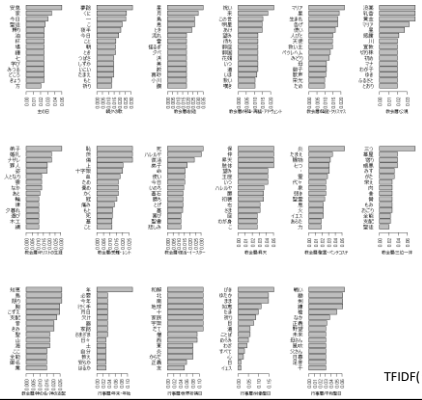
$$= \log\left(1 / \frac{\text{当該の単語を含む文書の数}}{\text{総文書数}}\right)$$

$$= \log\left(\frac{1}{\text{文書頻度}}\right)$$

- すべての文書に当該の語が含まれる場合、IDFはゼロになる
- 「当該の単語を含む文書の数」がゼロになると計算できないので、実用上 1 を足して計算することがある



TFIDF(トークン)



TFIDF(タイプ)

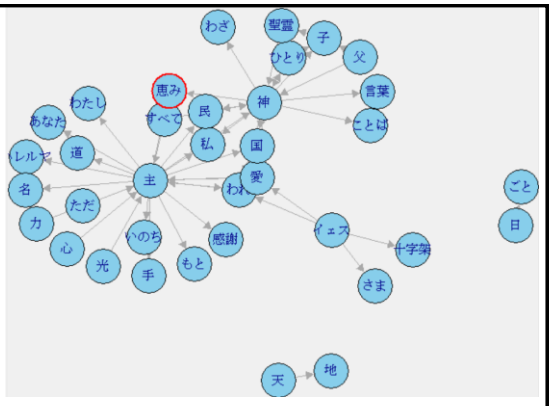
共起分析 co-occurrence analysis

- 共起をカウントする範囲
  - 文書
  - 章・節
  - 文
  - 窓関数
    - 前後n単位

  ※範囲が広くなるほどデータサイズが膨大になる
- カウントの仕方
  - Bag of words: 出現のみを考慮する
  - N-gram: 語順も考慮に入れる

N-gramを用いた共起分析

- 手続き
  - 対象とする単語群を決める（e.g. 品詞）
  - N-gramを抽出する
  - カウントする
  - 分析・可視化する
- 特徴
  - 順序・距離が保持される
    - 文章の空間的構造が比較的反映されている
      →距離的に近い組み合わせを抽出できる
  - 数が少なくすむ
  - 直近の共起関係しかわからない



『讃美歌21』全体の名詞のバイグラム(freq > 10)

★Bag of wordsを用いた共起分析

- 手続き
  - 対象とする単語群を決める(e.g. 品詞)
  - 文ごとに単語の共起を抽出する
    - Bag of wordsに基づいて、総組み合わせを抽出する
  - 集計する
  - 分析・可視化
- 特徴
  - 文内の距離・順序を保持しない
    - 空間的構造は反映されない
      →文内の距離に影響されない
  - 組み合わせが膨大になる
    - Bag of wordsの長さの二乗に比例する
  - 意味的なつながりの薄い組み合わせも拾ってしまう

この分析の持つ課題

- 単語の問題
  - 表記ゆれ
    - 同じ言葉の表記ゆれがある　e.g. 漢字/ひらがな
    - 対策
      - プログラムを使って検索・置換し、統一する
      - プログラム上で置換するのが難しい場合は、表記を統一したデータセットを作る
        - もともとのデータも残しておいて後で差分をとれるようにする
  - 単語の誤認識
    - 単語区切りの間違い、品詞カテゴリの間違い
    - 対策
      - （軽微な場合）読み込んだ後で修正する
      - 辞書を整備する
- 集計の問題
  - 集計方法によって結果は変わる
    - トークン/タイプ, ストップワード
  - 対策
    - 複数の集計方法を試してみて、結果が頑健であることを確認する

小まとめ

- データのクリーニング
  - ノイズを減らす、辞書を整備する
  - 分析しながら適宜データを洗練させていく
  - データ固有の知識が役に立つ
- 前処理/データハンドリング
  - テキストは様々な単位で分析できる
- 簡単な分析でも色々なことがわかる
  - 頻度の分析
  - TF-IDF
  - バイグラム

分析DEMO
②CCLI TOP 100のテキスト分析

CCLI TOP 100

- CCLI : Christian Copyright Licensing International
  - キリスト教徒の礼拝に関する歌、映像、ビデオ等の著作権を集中管理している機関。
  - CCLI TOP 100: 礼拝で使われているCWM楽曲の人気ランキングのトップ100
    - Spotifyに同名のプレイリストもある
  - Contemporary Worship Music (CWM)
    - ポップ・ミュージックと同様のスタイルのキリスト教音楽のジャンル
- データ：
  - 2021年1月8日時点でのCCLI TOP100に含まれる楽曲の歌詞

処理の方針

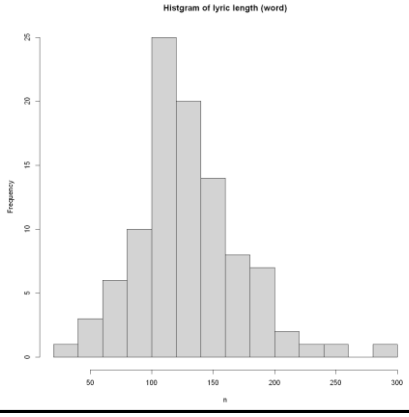
- 英語の歌詞の曲を分析する
  - ID42"Let Me Stay"は中国語なので除外する
- 分析に使うモジュール
  - 英語の文章は当然RMeCabでは分析できないので、tidytextパッケージを使う
  - tidytextではPOS(品詞)情報のタグ付けが面倒なので、stemmingで単語をまとめる
    - Snowball stemmerを使う

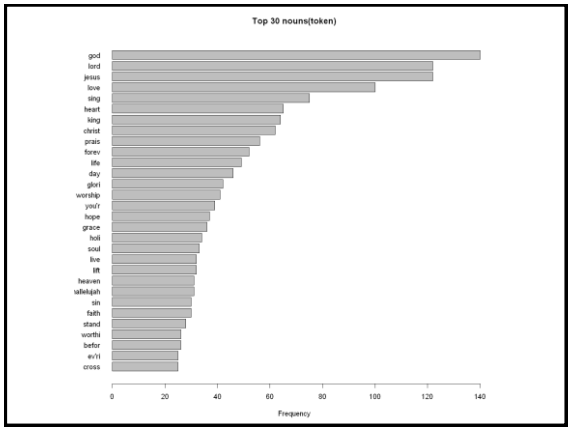
歌詞の分析にあたって（再掲）

- 歌詞：音楽に合わせて歌われるテキスト
  - 通常の散文ではない
    - 繰り返しのフレーズがある場合がある
      - 同じ単語の出現回数がかなり増える可能性がある
  - 音楽的構造を持つ
    - 形式：一般的なコラールはAAB形式
    - 音楽的要素・修辞との関係性も検討可能

分析の方針

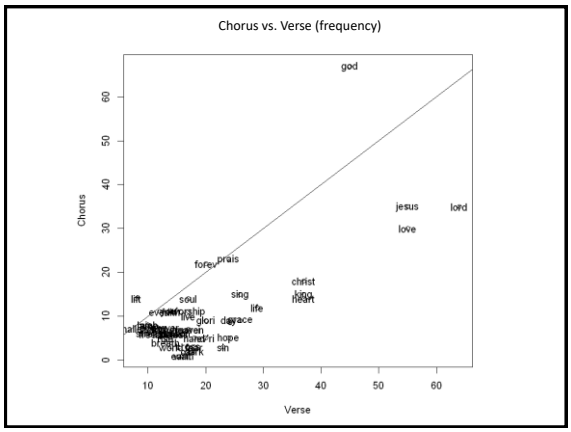
- 全体の特徴をつかむ
  - 全体の頻出語彙の抽出
- 楽曲構造を利用した分析
  - ポップミュージック：Verse-Chorus形式が多い
    - 山場であるChorusとVerseの対比的構造
  - 対比的構造が歌詞にも見られるか？
    - 単語の頻度の比較

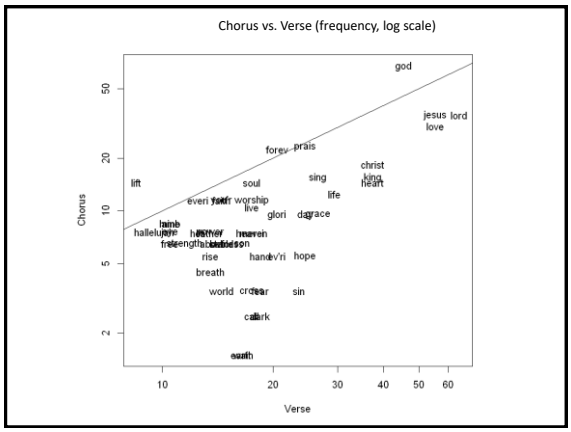




ポップミュージックの構造

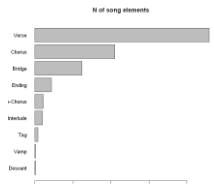
- Verse-Chorus形式
  - 山場であるchorusと、引き立たせるverseとの組み合わせ
    - 日本的な言い方をするとverseがAメロ、chorusがサビ
    - verseとchorusをつなぐbridgeなどといった要素を含むこともある
    - e.g. Beatles "Penny lane"
      - Verse 1 → Verse 2 → Chorus → Verse 3 → Solo → Chorus → Verse 4 → Verse 5 → Chorus → Coda(chorus)
- 32-bar形式
  - 構成：AABA
    - A部に力点がある
      - e.g. "Over the Rainbow"
- cf. コラール形式
  - 構成：AAB

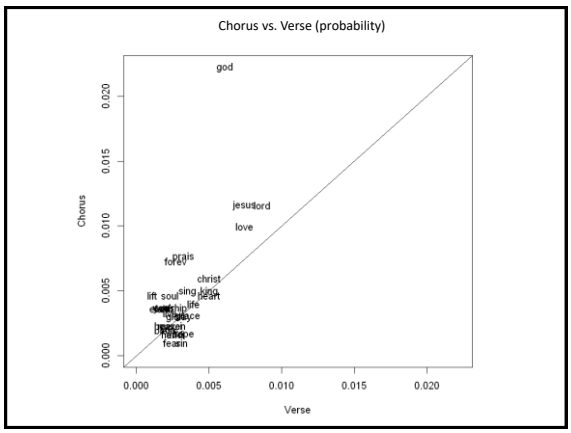


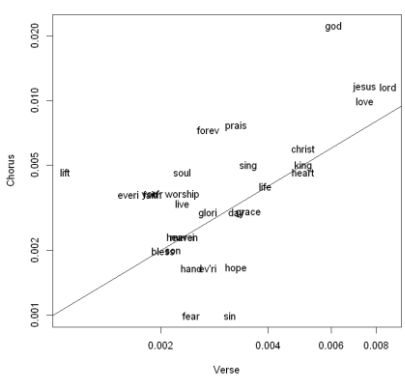


適切な比較？

- そもそもVerseの方がChorusよりも多い
  →単純な頻度による比較はできない
- 頻度をそれぞれの総単語数で割ることで確率に変換する





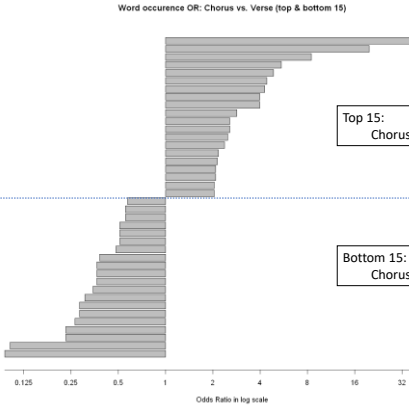


オッズ比

- オッズ(odds)：結果が生じる見込みの指標

$$Odds = \frac{p}{1-p}$$

- オッズ比(odds ratio: OR)
  - オッズ同士の比
  - 確率が小さい場合には確率の比(リスク比)のよい推定値になる



この分析の持つ課題

- 語彙の問題
  - Stemmingの問題
    - 語幹のみ使うことによる不正確さ
      - e.g. opera も operationも'oper'になってしまう
    - 品詞情報の欠落
    →品詞のタグ付けなどを試す必要
  - 否定表現
    - 単純に単語を抜き出しただけではそれがどのように使われているかまではわからない
    - 「Aをした/Aである」ではなく「Aをしなかった/Aでない」かもしれない
    →構文解析などといった処理を使う必要がある
- 楽曲構造の問題
  - それぞれの楽曲が実際にVerse-Chorus形式になっているかどうか確認していない(色々な形式を含むかもしれない)
  →実際の楽曲の形式を確認する必要

全体のまとめ

- テキストマイニング/計量テキスト分析
  - 自然言語であるものを量的研究の俎上に載せられる
  - テキストさえあれば研究できる
  →研究の幅が広がる
- 注意を払うべきこと
  - テキスト分析固有の罠もある
  →データの内容および背景知識、手法に関する知識を持つことで、罠にはまる可能性を減らせる

手法の可能性と限界を理解しつつ活用することで
意義ある知見を生み出すことができる