【AI開発道場】
自然言語処理

目次

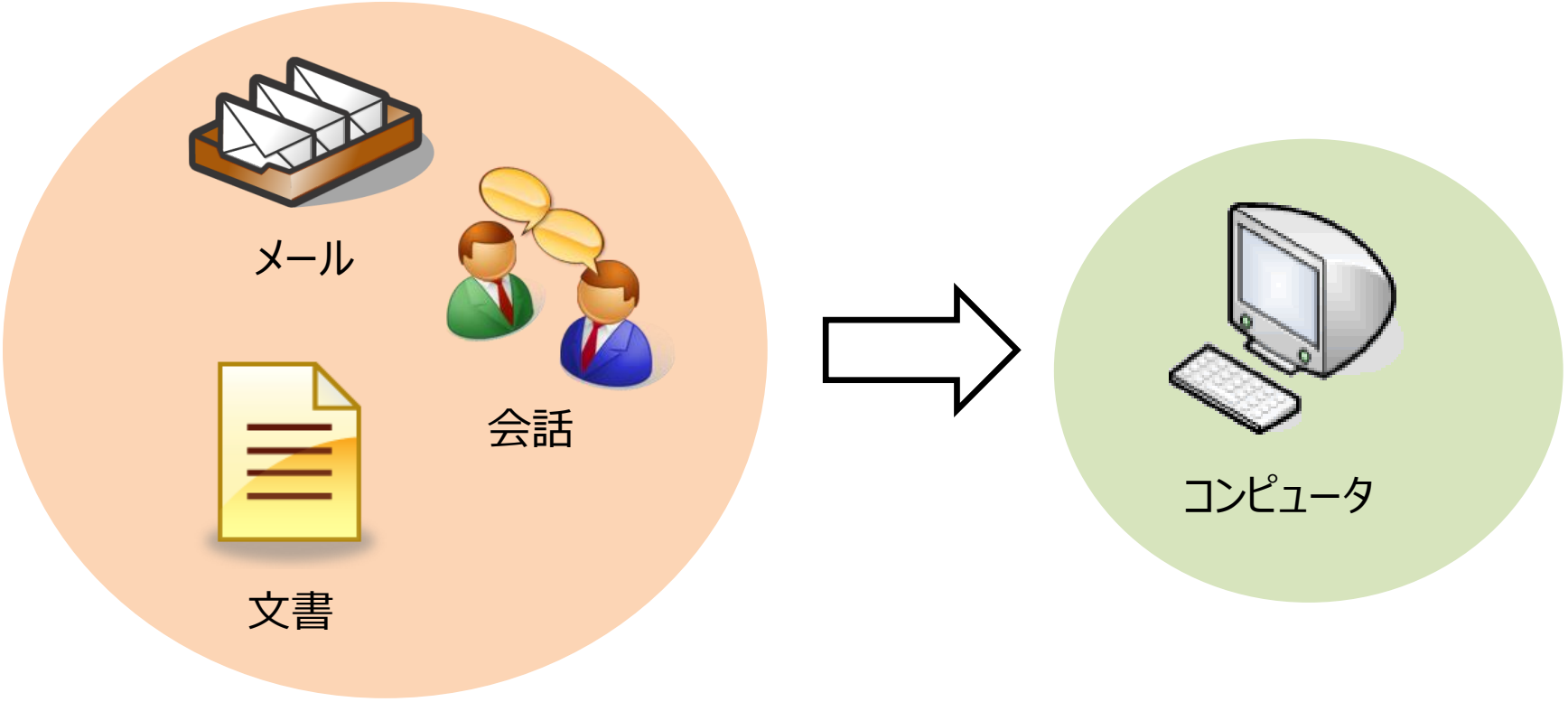
- 自然言語処理とは
- 基礎技術紹介
- 複合技術
- 活用例
- まとめ

目次

- 自然言語処理とは
- 基礎技術紹介
- 複合技術
- 活用例
- まとめ

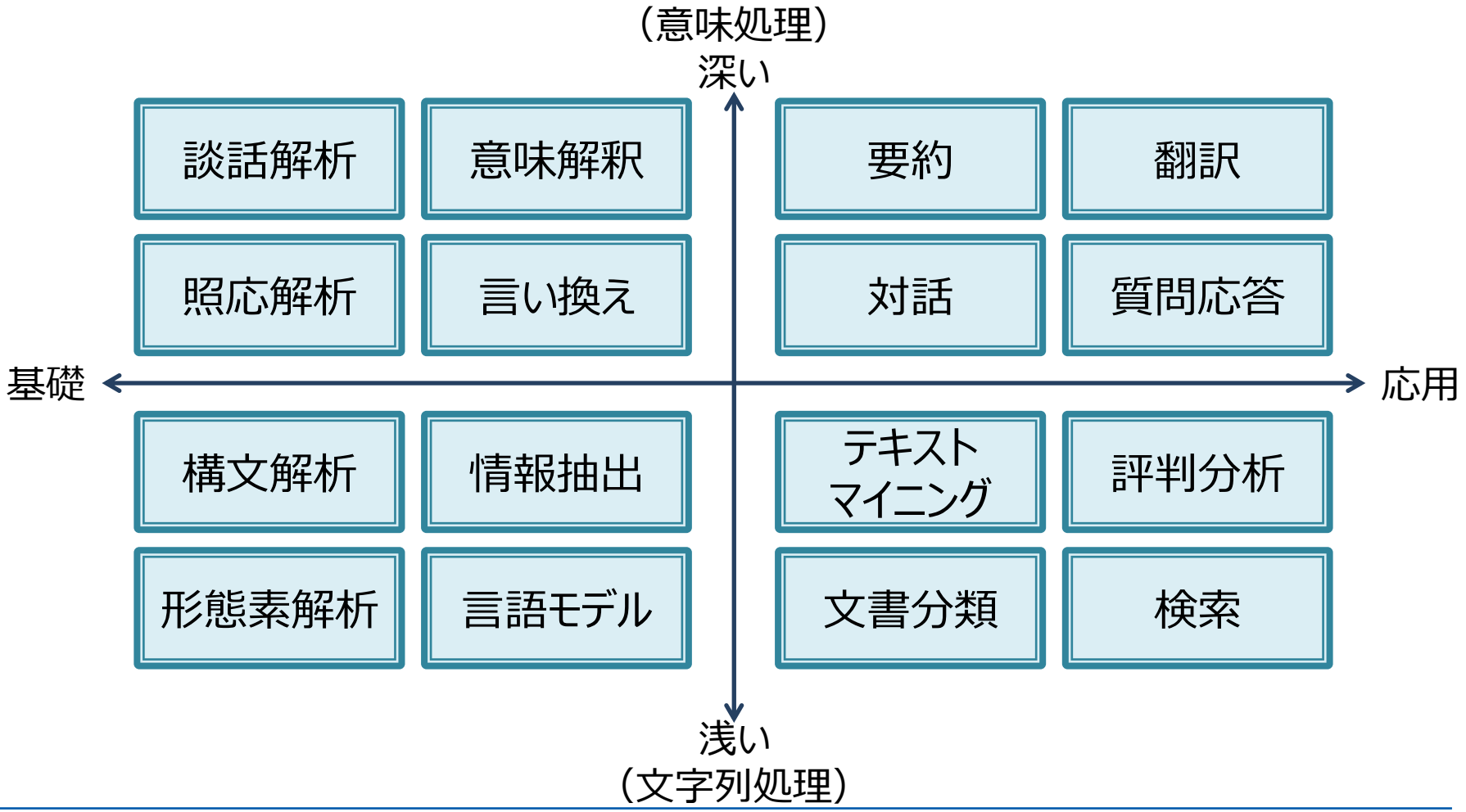
自然言語処理とは

- 人間が日常的に使っている日本語や英語等の自然言語を、コンピュータに処理させる一連の技術



自然言語処理とは

- 基礎から応用まで幅広い



自然言語処理のタスク

- 自然言語処理には様々なタスクがある
  - 文書分類タスクとその変形
  - 質疑応答
  - トークンに対する分類問題とその変形
  - 系列ラベリングとその変形
  - 系列変換・生成タスクとその変形
  - 対話
  - 言語モデル
  - 意味解析
  - 知識獲得
  - 文体変換
  - マルチモデルタスク

自然言語処理のタスク

- 文書分類タスクとその変形
  - 文書分類
  - 感情分析
    - 感情分析
    - ABSA(Aspect-Based Sentiment Analysis)
    - Subjectivity Analysis（主観/客観判断）
  - Stance Detection（肯定/否定判断）
  - 自然言語理解
    - 自然言語推論
    - 意味的類似度評価
    - 言い換え認識

自然言語処理のタスク

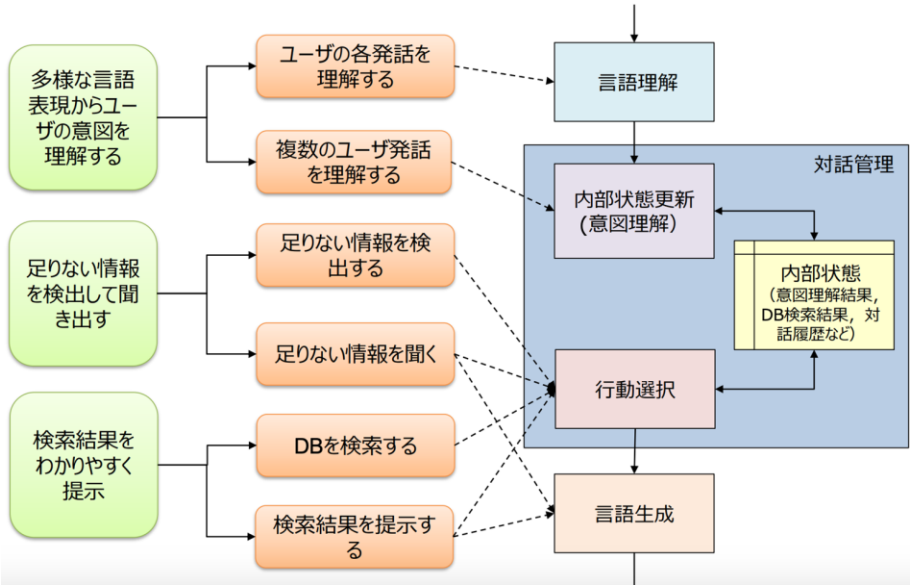
- 質疑応答
  - 択一式問題への回答
  - 機械読解
  - 対話形式の質問応答
  - Knowledge Base Question Ansewring
- トークンに対する分類問題とその変形
  - Missing Elements
    - Numeric Fused-Head（省略語の復元）
  - 語義曖昧性解消
    - 語義曖昧性解消
    - 語義推定

自然言語処理のタスク

- 系列ラベリングとその変形
  - 品詞タグ付けPOS(Part-of-speech) Tagging
  - 浅い構文解析 Shallow Syntax
  - 構文解析 Parsing
    - 句構文解析
    - 依存構造解析（係り受け解析）
      - 教師あり
      - 教師なし
  - 深い構文解析
    - 組み合わせ範疇文法
    - 組み合わせ範疇文法による構文解析
  - 意味役割付与（述語構造解析）
  - 共参照解析 Conference Resolution
  - 固有表現抽出 NER(Named Entity Recognition)
  - エンティティリンキング Entity Linling

自然言語処理のタスク

- 対話
  - 言語理解/意図理解
    - 形態素解析：Mecab,JUMAN++等
    - 対話行為タイプ推定: BoW, TF-IDF,SVM,Word2Vec等
    - スロット抽出：固有表現抽出等
    - 同義語辞書
  - 行動選択
    - DB検索等
  - 言語生成
    - テンプレート生成等
  - 参考URL



自然言語処理のタスク

- 言語モデル Language Modeling
  - 言語モデル
- 意味解析
  - 抽象的意味表現 AMR(Abstract Meaning Representation)
  - SQL Parsing
- 知識獲得など
  - Taxonomy Learning
  - Common Sense
  - 情報抽出 IE(Information Extraction)
    - Open Knowledge Graph Canonisalization

自然言語処理のタスク

- 文体変換
  - 文体変換 text Style Transfer
- マルチモデルタスク
  - 言語×画像
    - 画像のキャプション生成
    - VQA(Visual Question Ansewring)
  - 言語×データ
    - Data-to-Text Generation
  - 言語×音声
    - 音声認識

目次

- 自然言語処理とは
- 基礎技術紹介
- 複合技術
- 活用例
- まとめ

基礎技術

- 本日紹介する内容

テキスト整形

形態素解析

構文解析

分散表現

トピックモデル

テキスト整形

- 自然言語処理における基礎の基礎
- まずは以下のような整形処理を行う

半角全角	• 英数字は半角、カナは全角に統一する
大文字小文字	• どちらかに統一
改行コードの削除	• 改行コードが邪魔になることが多い
区切り文字によるデータ分割／統合	• csvやjsonを扱う場合に必要
不要文字（記号等）の削除	• 正規表現が強力
置換	• 正規表現が強力
検索	• 正規表現が強力
抽出	• 正規表現が強力

表記ゆれとその対策

- 表記ゆれとは、同音・同意味の語句について異なる文字表記が付されることである
  - 様々なパターンがある

漢字・仮名・カナの表記ゆれ

•例：「ネコ」「猫」「ねこ」

送り仮名の有無による表記ゆれ

•例：「引っ越し」「引越し」「引っ越」「引越」

異体字や代用漢字による表記ゆれ

•例：「斎藤」「斉藤」「齋藤」

同音異義語のよる間違い

- 例：「人口知能」と「人工知能」

固有名詞の通称・略称・カタカナ表記

- 例：「UNIQLO」と「ユニクロ」、「Yahoo!」と「ヤフー」

読みに由来する表記ゆれ

- 例：「ヴェネツィア」と「ベネチア」、「サーバ」と「サーバー」

誤字・脱字

- 例：「ディズニーランド」「デズニーランド」「でぃーずにランド」

省略による表記ゆれ

- 例：「ダイヤ」と「ダイヤモンド」

外来語の由来等による表記ゆれ

- 例：「ヴェニス」と「ヴェネツィア」

同じ意味を持つ語による表記ゆれ

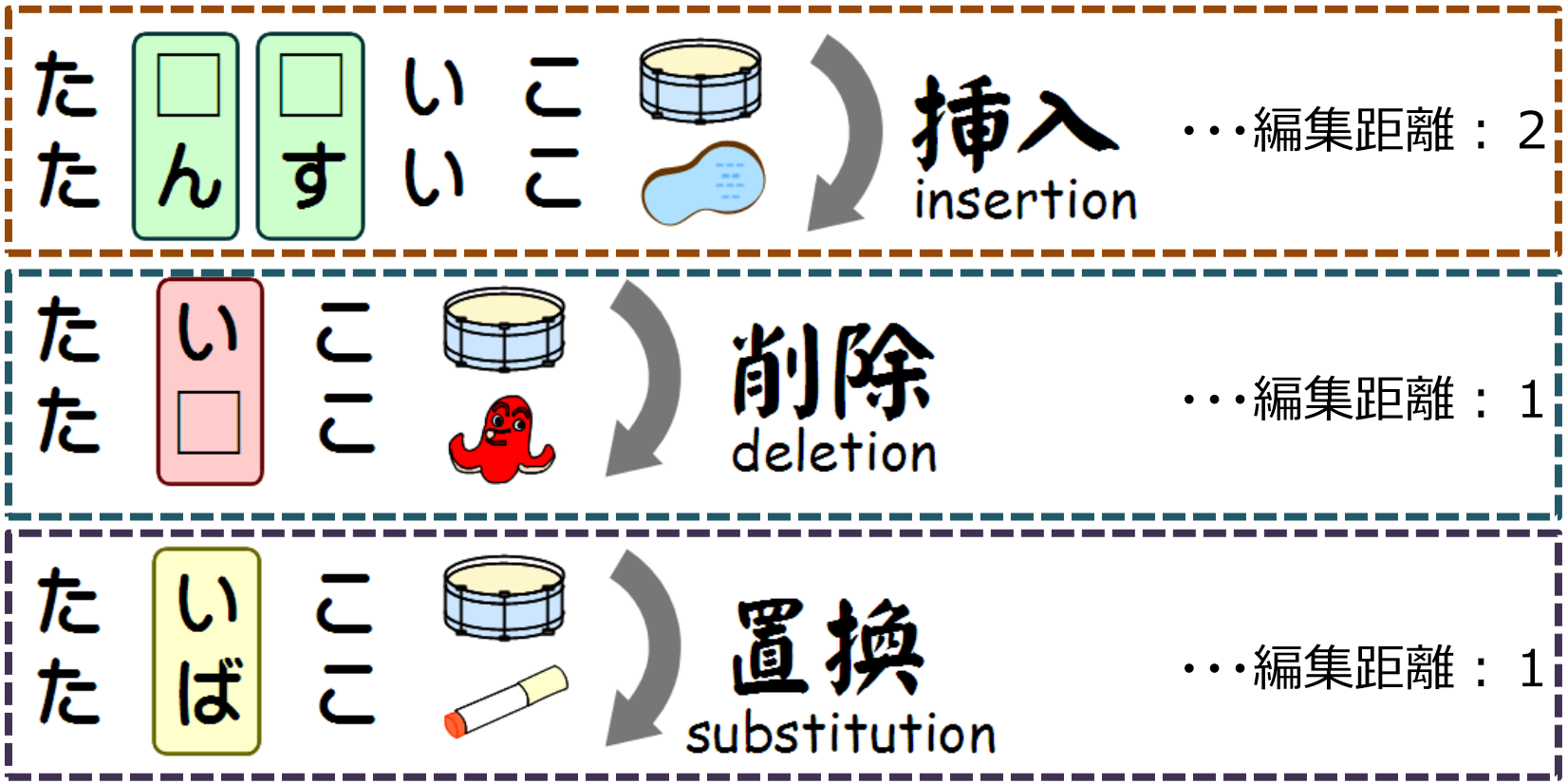
- 例：「同意」と「賛成」、「転居」と「移転」

表記ゆれとその対策

- 対策
  - 形態素解析
    - 同じ品詞・同じ読みは同一の単語とみなす
    - 固有名詞は辞書に登録する
  - 編集距離
    - 二つの文字列がどの程度異なっているかを示す距離
    - 一文字の挿入・削除・置換によって、一方の文字列を
      もう一方の文字列に変形するのに必要な手順の最終回数
    - ある程度距離が近いものは同じ単語とみなす
  - ベクトル化（Word2Vec）
    - 後述

【補足】編集距離（レーベンシュタイン距離）

- 挿入・削除・置換



表記ゆれとその対策

- マッピング

形態素解析

漢字・仮名・カナの表記ゆれ
- 例：「ネコ」「猫」「ねこ」

送り仮名の有無による表記ゆれ
- 例：「引っ越し」「引越し」「引っ越」「引越」

異体字や代用漢字による表記ゆれ
- 例：「斎藤」「斉藤」「齋藤」

同音異義語のよる間違い
- 例：「人口知能」と「人工知能」

固有名詞の通称・略称・カタカナ表記
- 例：「UNIQLO」と「ユニクロ」、「Yahoo!」と「ヤフー」

読みに由来する表記ゆれ
- 例：「ヴェネツィア」と「ベネチア」、「サーバ」と「サーバー」

誤字・脱字
- 例：「ディズニーランド」「デズニーランド」「でぃーずにランド」

省略による表記ゆれ
- 例：「ダイヤ」と「ダイヤモンド」

編集距離

外来語の由来等による表記ゆれ
- 例：「ヴェニス」と「ヴェネツィア」

ベクトル化

同じ意味を持つ語による表記ゆれ
- 例：「同意」と「賛成」、「転居」と「移転」

形態素解析

- ある文章・フレーズを「意味を持つ最小限の単位（形態素）」に分解すること
  - 例：すももももももものうちを形態素解析

```
すもももももものうち
すもも　名詞,一般,*,*,*,*,すもも,スモモ,スモモ
も　助詞,係助詞,*,*,*,*,も,モ,モ
もも　名詞,一般,*,*,*,*,もも,モモ,モモ
も　助詞,係助詞,*,*,*,*,も,モ,モ
もも　名詞,一般,*,*,*,*,もも,モモ,モモ
の　助詞,連体化,*,*,*,*,の,ノ,ノ
うち　名詞,非自立,副詞可能,*,*,*,うち,ウチ,ウチ
EOS
```

形態素解析

- 出力の詳細
  - 例：「形態素解析してみる」

書字形	発音形	語彙素読み	語彙素	品詞	活用型	活用形	語形	書字形基本形	語種
形態	ケータイ	ケイタイ	形態	名詞-普通名詞-一般			ケータイ	形態	漢
素	ソ	ソ	素	接尾辞-名詞的-一般			ソ	素	漢
解析	カイセキ	カイセキ	解析	名詞-普通名詞-サ変可能			カイセキ	解析	漢
し	シ	スル	為る	動詞-非自立可能	サ行変格	連用形-一般	シ	する	和
て	テ	テ	て	助詞-接続助詞			テ	て	和
みる	ミル	ミル	見る	動詞-非自立可能	上一段-マ行	終止形-一般	ミル	みる	和

形態素解析

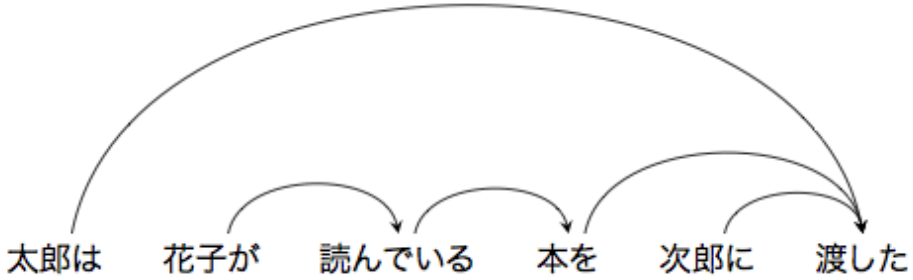
- 考慮点
  - 活用について
    - 文章で現れた形態素をそのまま使うか、基本形を使うか
    - 一般的には基本形を使用することが多い
  - 辞書について
    - 形態素解析をする際には、辞書を使用している
    - 辞書に載っていない単語は、正しく形態素解析されない
    - 辞書をカスタマイズする必要がある
  - 出力形式について
    - 形態素解析エンジンでは、出力形式を選択できる
      （前述のような標準出力や、分かち書き等）
    - 用途に合わせた出力形式を選択する

形態素解析

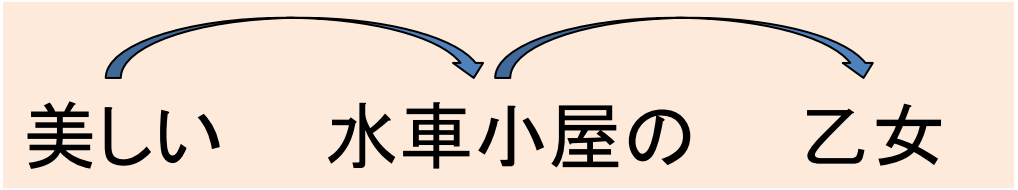
- 代表的なライブラリ
  - Mecabが有名で、多様な言語に対応
    - mecab本体とpythonラッパーが必要
    - デファクト
  - Janome
    - Pip installコマンドで一発導入可能
    - 精度はmecabからちょっと劣る

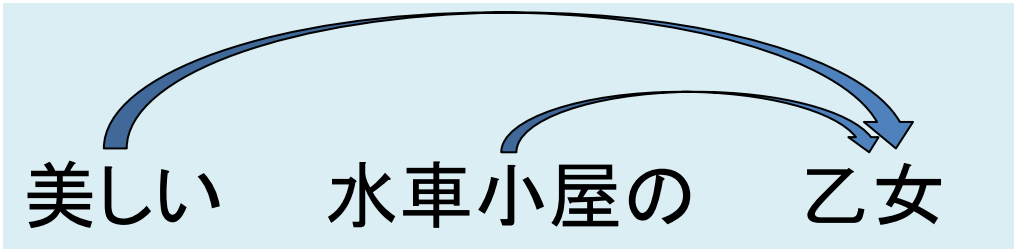
構文解析

- 語句の間にある、「修飾する」「修飾される」関係
  - 例：



  - 難しい例：





- Cabochaが有名

分散表現

- 分散表現とは
  - 自然言語をコンピュータが扱える形式に変換すること
  - 基本的には高次元のベクトル表現を利用

- 代表的な手法
  - Bag of Words
  - TF-IDF
  - Word2Vec

Bag of Words

- 自然言語処理で文章を扱う際の、最も基本的な手法の一つ
- 単語の出現頻度によって文書をベクトル化する

①：会場には車で行きます
⇒（"会場"；1，"車"：1，"行く"：1）

②：会場には自動車で行きます
⇒（"会場"；1，"自動車"：1，"行く"：1）

③：会場には自転車で行きます
⇒（"会場"；1，"自転車"：1，"行く"：1）

④：お店には自転車で行きます
⇒（"お店"；1，"自転車"：1，"行く"：1）

	①	②	③	④
会場	1	1	1	0
お店	0	0	0	1
自動車	0	1	0	0
自転車	0	0	1	1
行く	1	1	1	1
車	1	0	0	0

Bag of Words

- 考慮点
  - 形態素解析で単語に分解した後の処理
    - どの品詞（名詞・動詞等）を使うかは要考察
  - 単語の出現フラグか、出現回数か
    - 用途や組み合わせる手法によって使い分ける
  - 単語数が増えると、ベクトルも大きくなる
    - 出現する単語数が、そのままベクトルの次元数に
      ⇒ 計算が大変！
      ⇒Word2Vec（後述）
  - 表記ゆれ対策は別途する
    - Bag of Wordsでベクトル化する前に、
      同じ意味の単語は表記を統一する処理を入れる等

TF-IDF

- TF-IDFは、文書中の単語に関する重みの一種
- 主に情報検索や文章要約などの分野で利用
- 以下の二つの指標にもとづいて計算する
  - TF（Term Frequency）：単語の出現頻度
  - IDF（inverse document frequency）：逆文書頻度

- BoWの単語出現回数の代わりに、TF-IDFを用いることで、単語毎の重要度を加味した表現が可能になる

TF-IDF

- TF
  - それぞれの単語の文書内での出現頻度。たくさん出てくる単語ほど重要

$$TF_t = \frac{(\text{単語 } t \text{ の出現回数})}{(\text{文書内の単語の総出現回数})}$$

- IDF
  - それぞれの単語がいくつの文書内で共通して使われているかを表す。多くの文書で横断的に使われている単語は重要ではない、ということ

$$IDF_t = \log \frac{(\text{全文書の数})}{(\text{単語 } t \text{ を含む文書の数}(DF))} + 1$$

TF-IDF

- 計算例
  - 以下のような３つの文書でTF-IDFを計算してみる

文書A「私はオレンジとリンゴではリンゴが好きだ。」
文書B「私は以前は青森に住んでいたが、今は東京都に住んでいる。」
文書C「私は青森産のリンゴが好きだ。」

  - 形態素解析すると・・・

文書A { 私, オレンジ, リンゴ, リンゴ, 好き }
文書B { 私, 以前, 青森, 今, 東京都 }
文書C { 私, 青森, 産, リンゴ, 好き }

  - 文書Aから順に見ていく

TF-IDF

- 計算例
  - 文書Aについて
    - 「私」のTFを算出
      - 文書Aの全単語の頻度数は5であり
      - 「私」が出現する回数は1回

文書A { 私, オレンジ, リンゴ, リンゴ, 好き }
文書B { 私, 以前, 青森, 今, 東京都 }
文書C { 私, 青森, 産, リンゴ, 好き }

$$TF(\text{私}, \text{文書A}) = \frac{1}{5} = 0.2$$

    - 「私」のIDFを算出
      - 「私」が出現する文書数は3

$$IDF(\text{私}) = ln\frac{3}{3} + 1 = 1$$

TF-IDF

- 計算例
  - 文書Aについて

文書A { 私, オレンジ, リンゴ, リンゴ, 好き }
文書B { 私, 以前, 青森, 今, 東京都 }
文書C { 私, 青森, 産, リンゴ, 好き }

- 同様に文書Aのすべての単語に対してTF・IDFを計算する

単語	TF	IDF
私	0.200	1.00
オレンジ	0.200	2.10
リンゴ	0.400	1.41
好き	0.200	1.41

- したがって、文書AのTF-IDFは、

TF-IDF(文書A) ＝ { 私, オレンジ, リンゴ, 好き }
＝ { 0.200, 0.420, 0.564, 0.281 }

TF-IDF

- 計算例
  - 文書B、C
    - 同様に文書B、CについてもTF-IDFを計算すると、

文書A { 私, オレンジ, リンゴ, リンゴ, 好き }
文書B { 私, 以前, 青森, 今, 東京都 }
文書C { 私, 青森, 産, リンゴ, 好き }

TF-IDF(文書A) = { 私, オレンジ, リンゴ, 好き }
= { 0.200, 0.420, 0.564, 0.281 }

TF-IDF(文書B) = { 私, 以前, 青森, 今, 東京都 }
= { 0.200, 0.420, 0.281, 0.420, 0.420}

TF-IDF(文書C) = { 私, 青森, 産, リンゴ, 好き }
= { 0.200, 0.281, 0.420, 0.281, 0.281}

TF-IDF

- 計算例
  - 文書B、C
    - 同様に文書B、CについてもTF-IDFを計算すると、

文書A { 私, オレンジ, リンゴ, リンゴ, 好き }
文書B { 私, 以前, 青森, 今, 東京都 }
文書C { 私, 青森, 産, リンゴ, 好き }

TF-IDF(文書A) = { 私, オレンジ, リンゴ, 好き }
= { 0.200, 0.420, 0.564, 0.281 }

TF-IDF(文書B) = { 私, 以前, 青森, 今, 東京都 }
= { 0.200, 0.420, 0.281, 0.420, 0.420}

TF-IDF(文書C) = { 私, 青森, 産, リンゴ, 好き }
= { 0.200, 0.281, 0.420, 0.281, 0.281}

全ての文書で横断的に登場する
「私」という単語は低く

TF-IDF

- 計算例
  - 文書B、C
    - 同様に文書B、CについてもTF-IDFを計算すると、

文書A { 私, オレンジ, リンゴ, リンゴ, 好き }
文書B { 私, 以前, 青森, 今, 東京都 }
文書C { 私, 青森, 産, リンゴ, 好き }

TF-IDF(文書A) = { 私, オレンジ, リンゴ, 好き }
= { 0.200, 0.420, 0.564, 0.281 }

TF-IDF(文書B) = { 私, 以前, 青森, 今, 東京都 }
= { 0.200, 0.420, 0.281, 0.420, 0.420 }

TF-IDF(文書C) = { 私, 青森, 産, リンゴ, 好き }
= { 0.200, 0.281, 0.420, 0.281, 0.281 }

全ての文書で横断的に登場する「私」という単語は低く

他の文書で出現していない単語は高くなっている

Word2Vec

- その名前の通り、単語をベクトル化（数値化）して表現する手法
- コンセプトは、「同じ文脈の中にある単語は、お互いに近い意味を持っている」
  - 例：
    - 「彼は優秀な人材である」
    - 「彼女は優れた社員である」

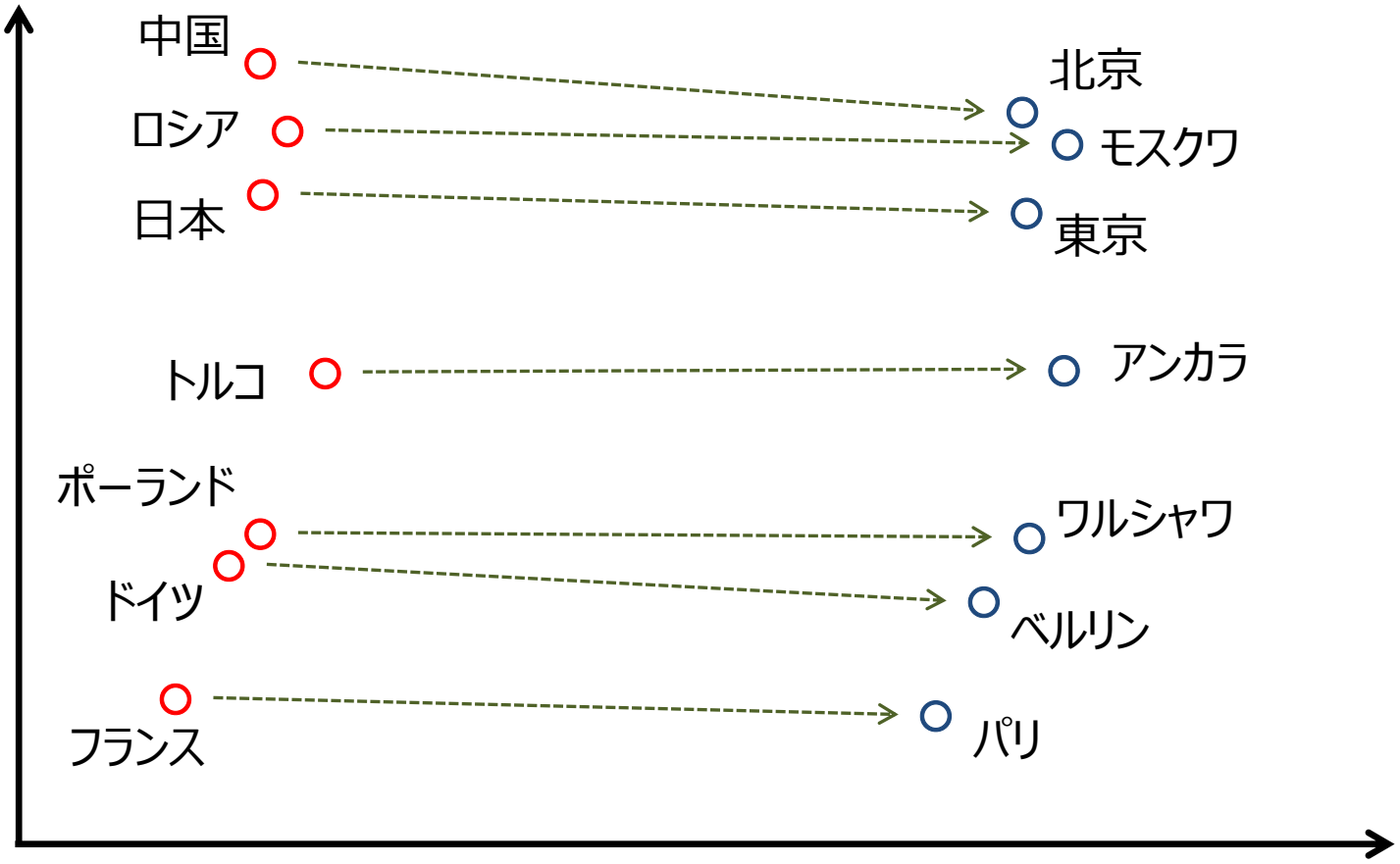
- 大量の文書データを学習させ、数値化する際の精度を高めていく

Word2Vec

- どのように学習するのか？
  - ある単語について、近くに出現した単語を調べる
    - 例：
      日本 → 政治，首相，国，貿易，選挙，税金
      中国 → 国家主席，税金，貿易，国，政治
  - 似たような単語が出現しているものは、
    似たような意味を持っているはず
    - 「日本」と「中国」は似た意味（国名）である
  - これを、すべての単語について調べていく

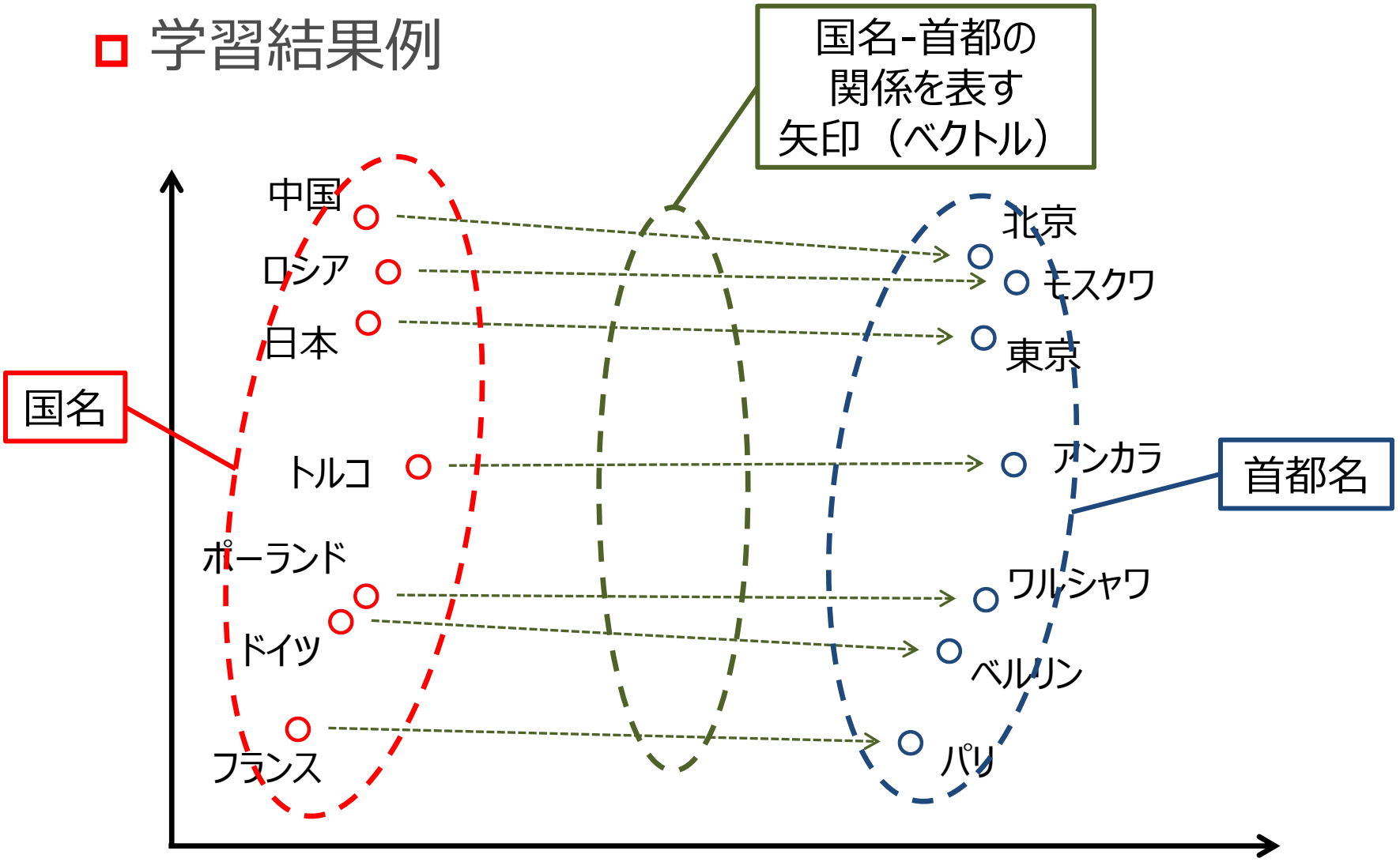
Word2Vec

- 学習結果例



Word2Vec

- 学習結果例



Word2Vec

- 使用例
  - 類似する単語の検索
    - 「日本」に類似する単語上位５単語の表示
      1. 「中国」 類似度：0.657877087593
      2. 「インド」類似度：0.605986833572
      3. 「韓国」 類似度：0.598236978054
      4. 「タイ」　類似度：0.584705531597
      5. 「シンガポール」　類似度：0.552470624447
  - 意味の演算
    - 「王様」-「男性」＋「女性」＝ ？

      A.「女王」
    - 「パリ」-「フランス」＋「イタリア」＝ ？

      A.「ローマ」

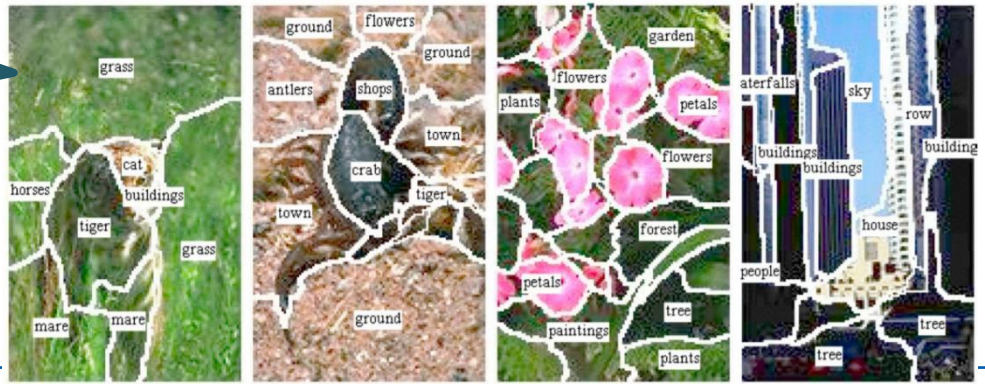
トピックモデル

- トピックモデル
  - トピック：ジャンル、話題、分野のようなもの
  - トピックモデル：
    データの背後にある隠れた「トピック」を推定する

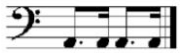
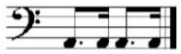
文書

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

画像



音楽

QUERY	RETRIEVED SONGS
come on, come on, get down	*Erksine Hawkins – Tuxedo Junction* *Moby – Bodyrock* *Nine Inch Nails – Last* *Sherwood Schwartz – 'The Brady Bunch' theme song*
	*The Beatles – Got to Get You Into My Life* *The Beatles – I'm Only Sleeping* *The Beatles – Yellow Submarine* *Moby – Bodyrock* *Moby – Porcelain* *Gary Portnoy – 'Cheers' theme song* *Rodgers & Hart – Blue Moon*
	
come on, come on, get down	*Moby – Bodyrock*

トピックモデル

- LDA
  - トピックモデルのデファクトスタンダード
  - トピックによって、出現する単語の確率が異なる
    - 例：
      - 政治 → 国・議会・選挙・政党 etc
      - スポーツ → 野球・リーグ・勝ち点・予選 etc
  - 文書は複数のトピックの組み合わせであると仮定
    - 例：
      - 文書A → 政治60%、スポーツ30%、その他10%
      - 文書B → スポーツ70%、科学20%、その他10%
  - 大量の文書データから、
    各トピックの単語の出現確率を推定したい

LDA

- LDAの実行結果例

Topic 1	Topic 2	Topic 3	Topic 4
先,後,#,歩,銀,四, 五,六,同,二,飛, 八,成,玉,七,三, 金,九,桂,角,と, 谷川,が,た,手,は, 丸山,一,香,の,で, 局,図,戦,段	の,号,事故,機,が, た,に,安全,#,部分, を,原発,原因,は, 基,水,運転,装置, 爆発,器,原子力, 炉,作業,し,燃料, で,漏れ,発生,と, 配管,原子,ガス	#,勝,敗,戦, イチロー,日,回, リーグ,大リーグ, マリナーズ,新庄, 試合,安打,点,ス, で,手,共同,メッツ, 外野,は,大,投手, 第,米,の,打席, ソックス, ヤンキース,記録, バックス,打率, ニューヨーク	研究,細胞, 遺伝子,移植, の,治療,物質, 教授,を,患者, 科学,脳,医療, 病院,ローン, ヒト,実験,薬, グループ,遺伝, が,臓器,体,ク, 病,する,に,学会, さ,DNA,開発, 臨床,人間,神経

LDA

- LDAの実行結果例

トピック数は、
あらかじめ定義する

Topic 1	Topic 2	Topic 3	Topic 4
先,後,#,歩,銀,四, 五,六,同,二,飛, 八,成,玉,七,三, 金,九,桂,角,と, 谷川,が,た,手,は, 丸山,一,香,の,で, 局,図,戦,段	の,号,事故,機,が, た,に,安全,#,部分, を,原発,原因,は, 基,水,運転,装置, 爆発,器,原子力, 炉,作業,し,燃料, で,漏れ,発生,と, 配管,原子,ガス	#,勝,敗,戦, イチロー,日,回, リーグ,大リーグ, マリナーズ,新庄, 試合,安打,点,ス, で,手,共同,メッツ, 外野,は,大,投手, 第,米,の,打席, ソックス, ヤンキース,記録, バックス,打率, ニューヨーク	研究,細胞, 遺伝子,移植, の,治療,物質, 教授,を,患者, 科学,脳,医療, 病院,ローン, ヒト,実験,薬, グループ,遺伝, が,臓器,体,ク, 病,する,に,学会, さ,DNA,開発, 臨床,人間,神経

実際には、それぞれの単語に
出現確率が付いている

LDA

- LDAの実行結果例

Topic 1
先,後,#,歩,銀,四,
五,六,同,二,飛,
八,成,玉,七,三,
金,九,桂,角,と,
谷川,が,た,手,は,
丸山,一,香,の,で,
局,図,戦,段

将棋

Topic 2
の,号,事故,機,が,
た,に,安全,#,部分,
を,原発,原因,は,
基,水,運転,装置,
爆発,器,原子力,
炉,作業,し,燃料,
で,漏れ,発生,と,
配管,原子,ガス

原子力発電所

Topic 3
#,勝,敗,戦,
イチロー,日,回,
リーグ,大リーグ,
マリナーズ,新庄,
試合,安打,点,ス,
で,手,共同,メッツ,
外野,は,大,投手,
第,米,の,打席,
ソックス,
ヤンキース,記録,
バックス,打率,
ニューヨーク

メジャーリーグ

Topic 4
研究,細胞,
遺伝子,移植,
の,治療,物質,
教授,を,患者,
科学,脳,医療,
病院,ローン,
ヒト,実験,薬,
グループ,遺伝,
が,臓器,体,ク,
病,する,に,学会,
さ,DNA,開発,
臨床,人間,神経

生命科学

目次

- 自然言語処理とは
- 基礎技術紹介
- 複合技術
- 活用例
- ツール・ライブラリ
- まとめ

応用技術

- テキスト要約
- 翻訳
- 対話
- テキスト感情分析

テキスト要約

- テキスト要約
  - 文章を要約するタスク
  - 以下のような観点で分類される
    - 入力方式
      - Single document：単一の文書
      - Multi-document：複数の文書
    - 条件付け
      - Generic：特に指示を出さない
      - Domain specific：特定の分野に特化
      - Query-based：与えられた質問やキーワードに対する要約
    - 要約範囲
      - Informative：重要な内容を網羅
      - Indicative：文章を読むべきか判断する情報を示す。本や映画のレビュー等
    - 出力スタイル
      - Keyword：キーワードのみ
      - Headline：ニュース記事のヘッドラインのように、一行のみ出力
      - その他指定された長さの要約文
    - 要約手法
      - Extractive：元の文章から抽出
      - Abstractive：人による要約のように文章の言い換えを利用する

テキスト要約

- アプローチ
  - 抽出型
    - 要約対象の文章の中から、重要と思われる分を抽出して要約を作成
    - メリット
      - 全く内容がつかめない要約になる可能性が低い
      - 文法的におかしな要約にならない
    - デメリット
      - 抽象化や言い換え、読みやすくするための接続しの使用ができない
  - 抽象型
    - 文章の意図をくみ取ったうえで、適切な要約を作成
    - メリット
      - より自由な要約を作成できる
      - 要約の長さに関する自由度が大きい
    - デメリット
      - 自然な文章を生成することが難しい

テキスト要約

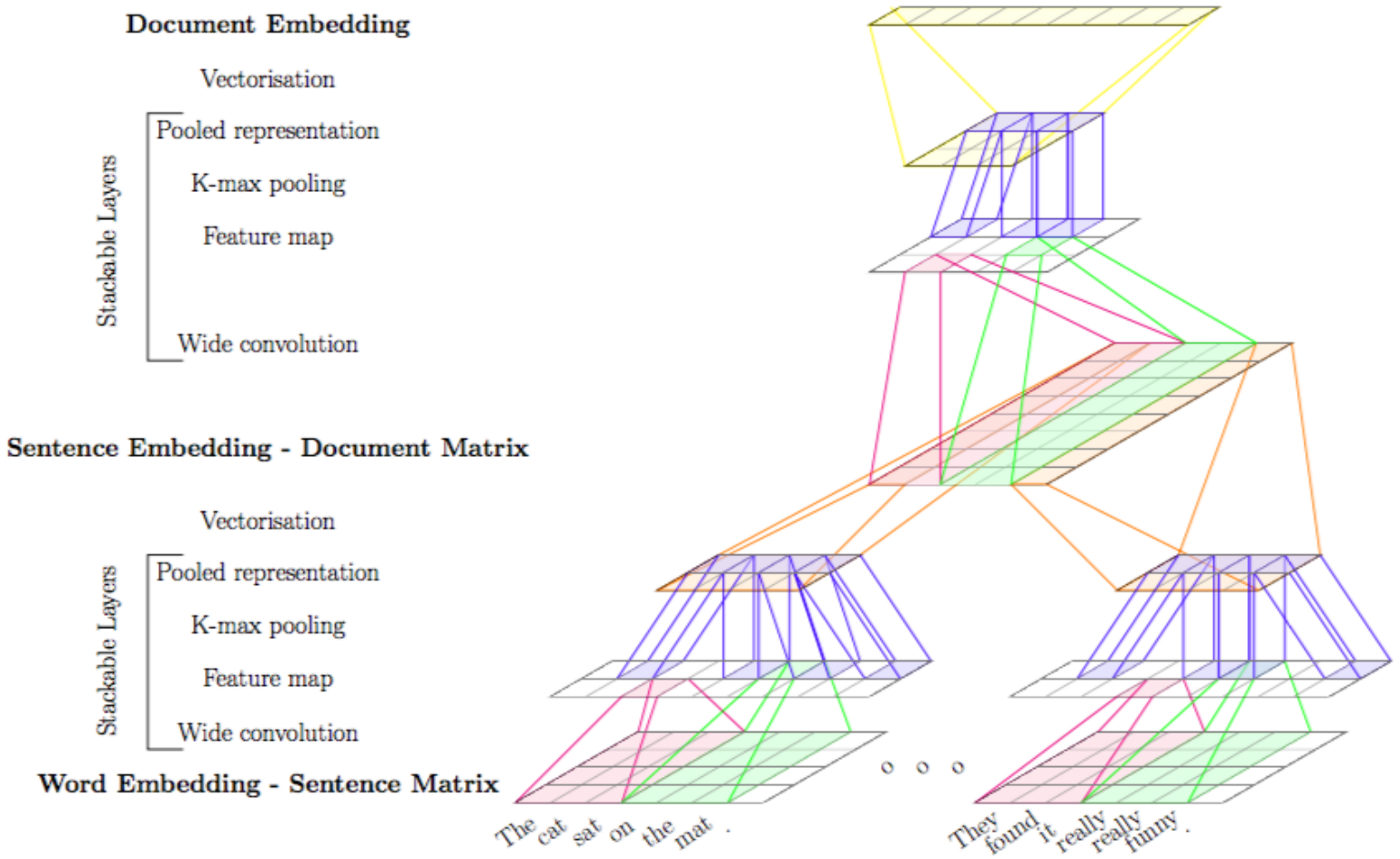
- 代表的な手法
  - 古典的な手法
    - 情報の出現頻度
      - 重要な情報ほど高い頻度で出現すると仮定し、キーワードやキーコンセプトを抽出
      - それらを多く含むセンテンスを高く評価
    - センテンス間の類似度
      - 他のセンテンスと類似するセンテンスはその文書の核となる情報を持っていると仮定
      - 検索エンジンで用いられるPage Rankアルゴリズムなど、文章のグラフ構造を用いる
    - その他
      - title feature: : タイトルに含まれる単語を含むセンテンスを高く評価
      - sentence position : 初めや最後のパラグラフに重要なコンセプトが含まれていると仮定
      - keyword frequency : Query-based要約の場合、キーワードを多く含むセンテンスを高く評価

テキスト要約

- 代表的な手法
  - 深層学習を用いた手法
    - 抽出型
      - CNN
      - CNN + Seq2Seq + Attention
    - 抽象型
      - ベースラインSequence-to-Sequence + Attention
      - Pointer-Gen
      - Pointer-Gen + Coverage

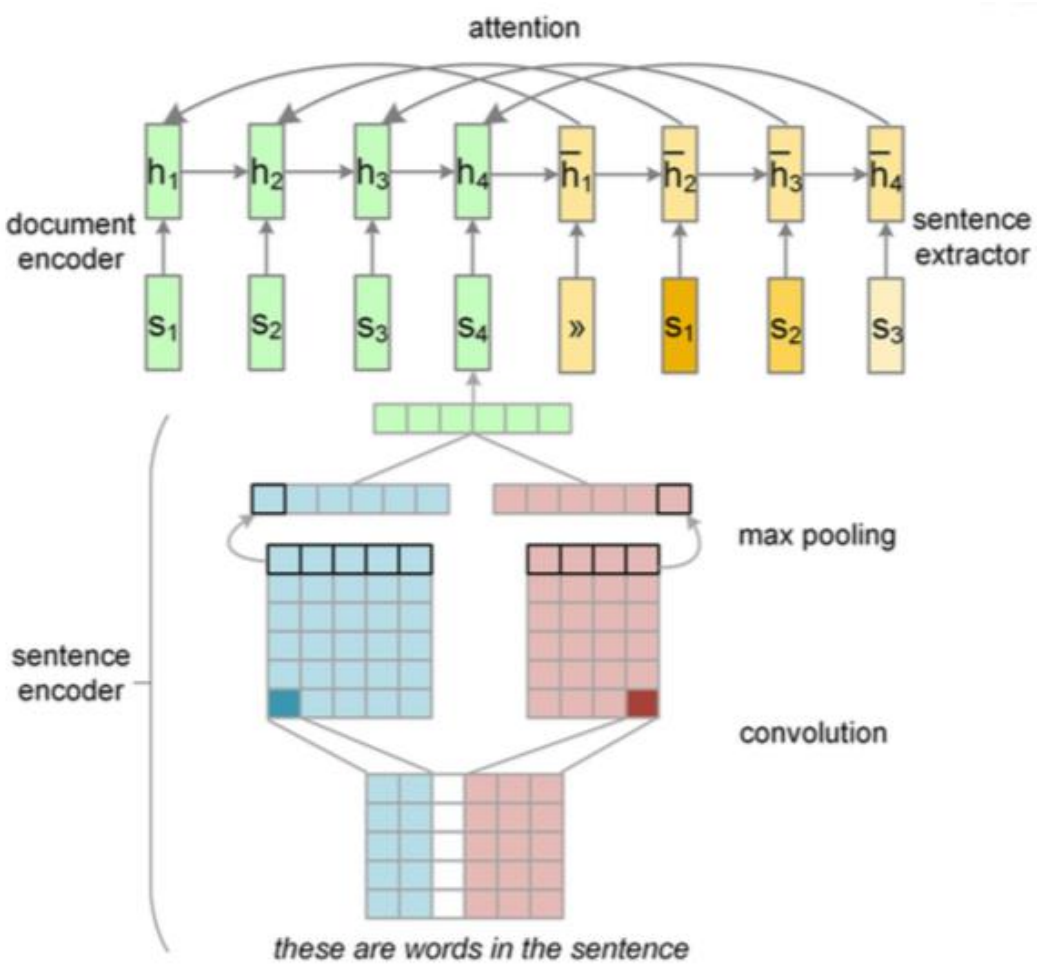
テキスト要約

- CNN


Document Embedding
Vectorisation
Pooled representation
K-max pooling
Feature map
Wide convolution
Stackable Layers
Sentence Embedding - Document Matrix
Vectorisation
Pooled representation
K-max pooling
Feature map
Wide convolution
Stackable Layers
Word Embedding - Sentence Matrix
The cat sat on the mat .
They found it really really funny .

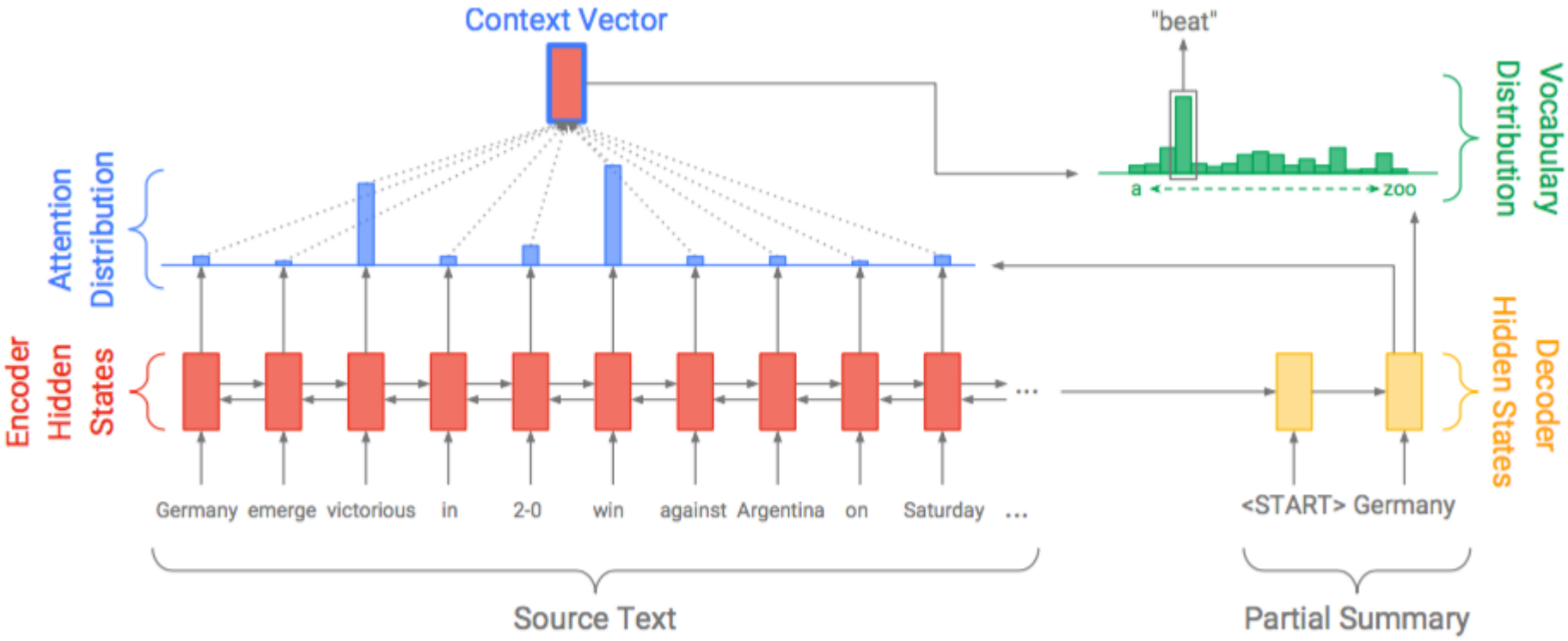
テキスト要約

- CNN + Seq2Seq + Attention



テキスト要約

- ベースラインSequence-to-Sequence + Attention

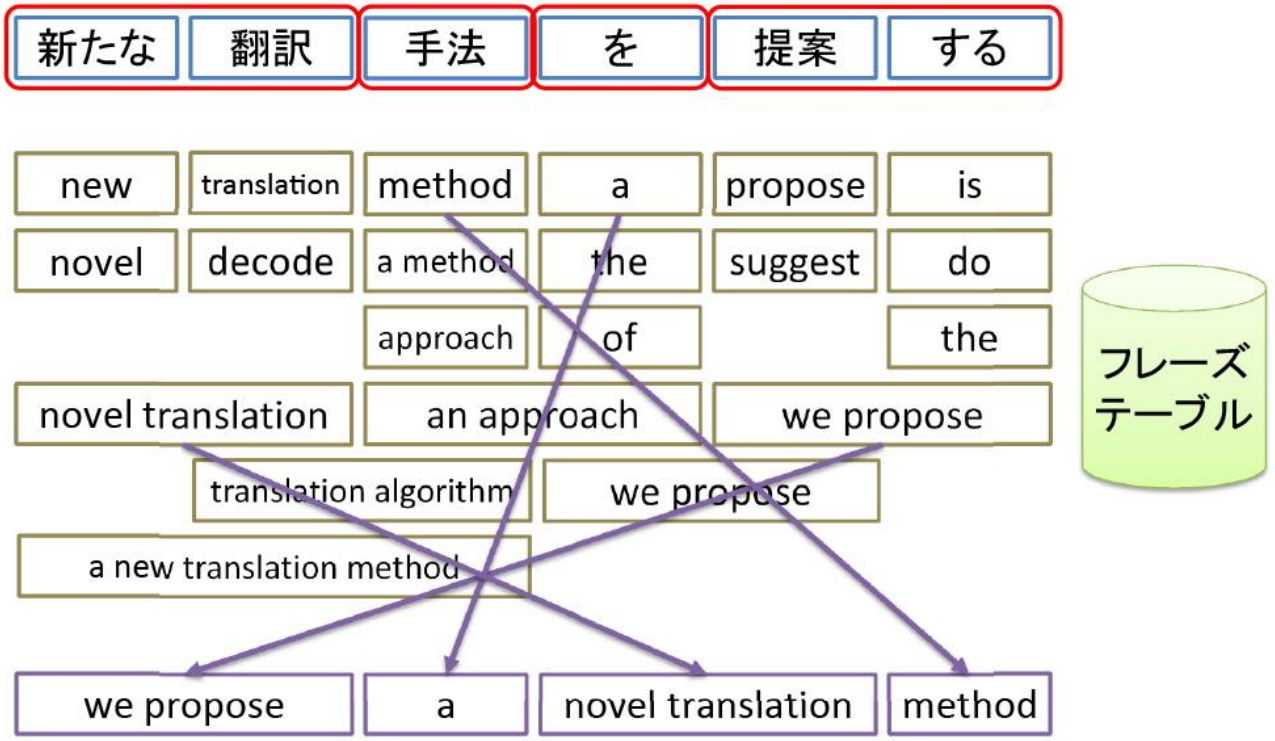


機械翻訳

- 言語翻訳
  - コンピュータを利用して、ある言語から別の言語に翻訳する技術
- アプローチ
  - ルールベース型
    - ルールと辞書を定義して翻訳する
    - 正確かつ揺らぎのない翻訳ができ、訳文の根拠を説明可能
    - ルールと辞書が膨大
  - 統計翻訳型
    - 原文と訳文のセットである「対訳コーパス」を基に、機械学習でに翻訳を行う
    - コンピュータが自動で学習するため手がかからない
    - コーパスの準備、計算コスト、訳文の根拠を説明不可
  - ニューラル翻訳型
    - 統計翻訳型にニューラルネットワークを取り入れたもの
    - 統計翻訳型とメリット・デメリットは変わらないが、制度が飛躍的に向上

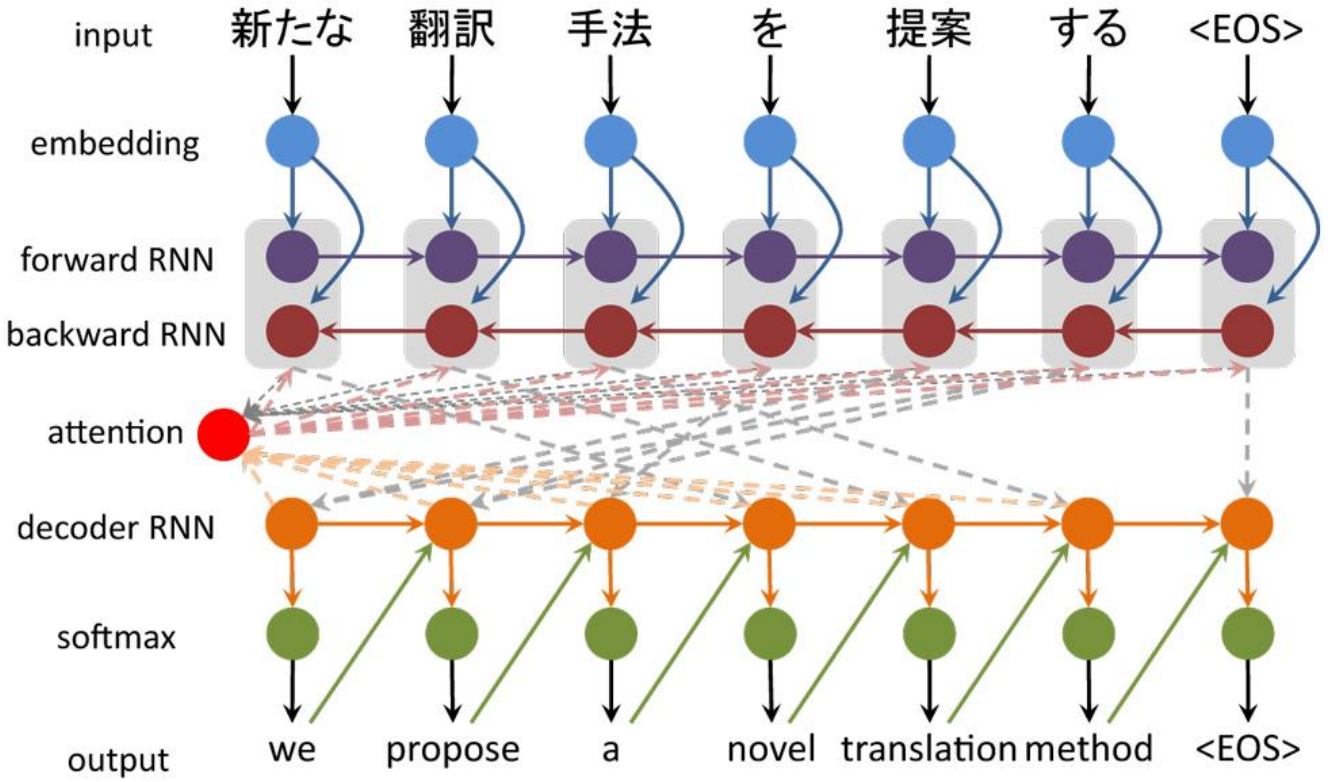
機械翻訳

- 統計翻訳型
  - 入力文を小さな部分に分割し、部分ごとに目的言語に置き換える
  - すべての部分が置き換えられたら翻訳が終了



機械翻訳

- ニューラル機械翻訳
  - 概略図



attentionより上がencoderで，下がdecoder

機械翻訳

- ニューラル機械翻訳
  - 大きく分けて以下の3つのパーツで構成されている
  - エンコーダー
    - 入力文を実数値の集合であつベクトル表現に符号化する機構
  - アテンション
    - 出力するべき単語を決定する際に、符号化された入力文のどこに注目すべきかをコントロール機構
  - デコーダー
    - 入力文とアテンションを基に、出力文を生成する機構

機械翻訳

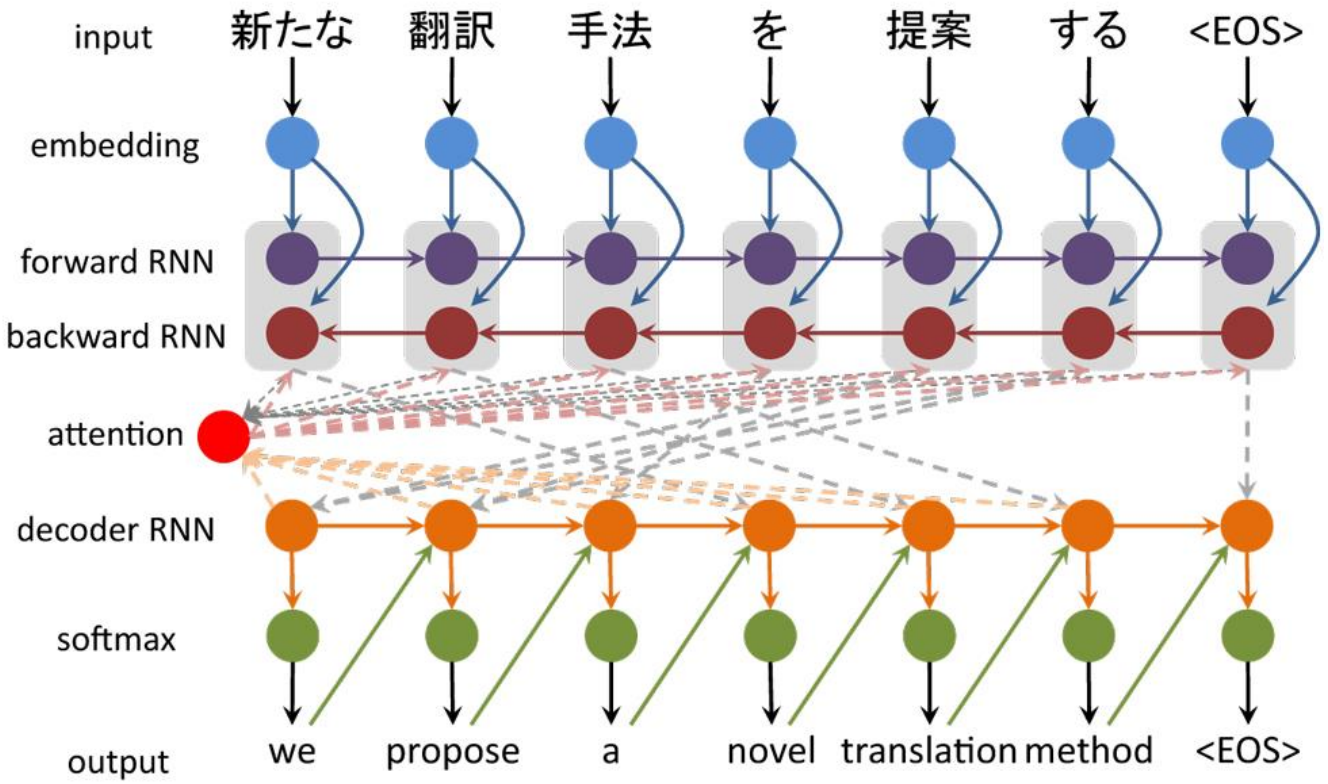
- エンコーダー
  - 単語の分散表現を行う機構（embedding）
  - Forward RNNとbackward RNNで、前後の単語の情報も考慮
  - 2つのRNNで生成された２つのベクトルをつなげたものを、各単語の分散表現として利用
- アテンション
  - 単語を翻訳するにあたり、注目すべき箇所を判断する機構
  - 入力文すべてに対し、注目すべき単語を確率的に計算し、入力分散表現の重み付き和（コンテキストベクトル）を計算する
  - エンコーダーとデコーダーのみ（Seq2Seq）では、単語の数が増えるとうまく翻訳できないが、アテンションを行うことによって、長文にも対応可能
- デコーダー
  - コンテキストベクトルと、一つ前に出力した単語から、次の単語を出力する
  - ソフトマックス関数により、最も確率の高い単語を出力する

対話

- 対話
  - 入力文章に対して最適な応答文を生成して出力する
- アプローチ
  - ルールベース型
    - 「Aという語句が含まれていたらBと返答する」というルールを大量に定義したもの
    - ELIZA
  - 機械学習型
    - ルールベース型に機械学習を組み合わせたもの
  - 深層学習型
    - ニューラルネットワークを用いて、入力から出力までを一気通貫で行う

対話

- 深層学習型
  - 概略図



attentionより上がencoderで，下がdecoder

対話

- 深層学習型
  - 機械翻訳と一緒？
    - ネットワーク構造はほとんど一緒
    - 学習データとするコーパスが異なる
  - 機械翻訳のコーパス
    - 入力
      - 原文
    - 出力
      - 原文を多言語に翻訳した文章
  - 対話のコーパス
    - 入力
      - 文章
    - 出力
      - 入力文章に対する返答

テキスト感情分析

- テキスト感情分析
  - 入力文章から、意見・感情・態度を推論するタスク
  - 比較的新しいタスク
- アプローチ
  - ルールベース
    - 手動で定義された一連のルールに基づいて感情分析を行う
  - 機械学習型
    - ルールベース型に機械学習を組み合わせたもの
  - 深層学習型
    - ニューラルネットワークを用いて、入力から出力までを一気通貫で行う

目次

- 自然言語処理とは
- 基礎技術紹介
- 複合技術
- 活用例
- まとめ

活用例

- AIりんな
  - https://www.rinna.jp/
  - 日本Microsoftが開発した会話AIボット
  - 自然言語処理アルゴリズムでWord2VecやTFIDFなど利用(当時)
- AIさくらさん
  - https://tifana.ai/
  - 接客AI
- ポケトーク
  - Google AI(音声認識,翻訳,音声合成)等を利用
- スマートスピーカー
  - Google Home, Amazon Alexa
- 検索エンジン
  - ElasticSearchではkuromojiの形態素解析等行う
- Asales
  - https://stockmark.co.jp/product/asales/
  - 東大発のAIベンチャー企業「ストックマーク社」が開発
  - 商談情報からニーズを抽出、提案書レコメンド
  - Googleが2018年に発表した自然言語処理モデル「BERT」を活用

演習

- 自然言語アプリの企画
  - 本日紹介した自然言語処理の技術を利用してアプリ案を考察する
  - アプリ案として、以下内容を含めること
    - アプリの目的
      - 何をするアプリなのか
    - アプリの機能
      - 目的を達成するために必要な機能
    - 入出力イメージ
      - 何を入力とし、どのような出力をするのか
    - 処理フロー
      - 入力を受け取って出力を行うまでの処理フロー
    - モデル作成フロー
      - 機械学習概論を参考にモデル作成フローを考察する
  - フォーマットは自由
  - 提出先：
    - ファイル ＞ AI概論_提出用フォルダ ＞ 自然言語処理 ＞ 演習
    - ファイル名：自然言語処理_名前.拡張子

目次

- 自然言語処理とは
- 主要技術紹介
- 複合技術
- 活用例
- まとめ

まとめ

- 自然言語処理とは、人間の言語をコンピュータに処理させる技術である
- 特に日本語を取り扱う場合、形態素解析や分散表現といったテキストデータの前処理が重要
- 現在は自然言語処理における様々な分野で、深層学習を用いた手法が考案されている

参考資料

- 自然言語処理の仕組みと手順
- 深層学習による自然言語処理の研究動向
- リクルート式　自然言語処理技術の適応事例
  - Word2vecで検索レコメンド
  - Doc2vecを使った文書要約