Vol.2020-AAC-13 No.3
2020/8/29

OCR の文字の確率と事前学習済み BERT の MASK の候補を組み合わせた後処理での認識率の改善

佐藤文一[†1]　喜連川優[†2]

概要：2019 年読書バリアフリー法が施行され，視覚障害者の読書環境の整備の推進が行われてきている．視覚障害者にとって情報障害の解消のためにも，OCR（光学文字認識）は重要な技術であり，更なる文字認識率の向上が望まれている．本論では，テキストデータを画像に変換し，OCR を行い，正解と OCR の結果のパターンマッチングにより，その認識率をすべて自動で行えるテスト環境を構築した．これにより，Wikipedia から約 36 万タイトル，総文字数約 4.3 億個から，個々の文字の認識率と何の文字にどのくらいの確率で誤っているかの統計データを得た．この OCR の誤り率と，事前学習済み BERT を使って単語を MASK し，その単語を推測して得られた候補とを組み合わせることにより，OCR の誤りの修正のテストを行った．事前学習済み BERT は 3 種類で行ったので，その結果を報告する．

キーワード：OCR，情報障害，BERT

Improvement of OCR recognition rate in post-processing by combining OCR character probability and pre-trained BERT MASK candidate

FUMIKAZU SATO[†1]　MASARU KITSUREGAWA[†2]

１．はじめに

「視覚障害者等の読書環境の整備の推進に関する法律」（通称：読書バリアフリー法）が 2019 年 6 月に公布・施行され，視覚障害者の読書環境の整備が行われ始めている．障害者の学習に対しては，2008 年に「障害のある児童及び生徒のための教科用特定図書等の普及の促進等に関する法律」（通称：教科書バリアフリー法）が成立し，視覚障害の小中学校の生徒に対しては拡大・点字教科書の提供が制度化された．また，障害者差別解消法の合理的配慮により，2016 年から大学の図書館においては，視覚障害者に対して図書のテキスト化のサービスが開始された．このように，障害者への情報アクセシビリティを向上するための施策が施行されてきているが，視覚障害者の情報障害に対しては，まだ克服すべき課題が多い．

視覚障害者に取って情報障害の解消のためにも，OCR（光学文字認識）は重要な技術であり，更なる文字認識率の向上が望まれている．たとえば，テキスト化のサービスでは，OCR の文字誤りの校正を人手で行っているため，かなりの期間が必要であり，高コストである．日本点字図書館でのテキスト DAISY 化でも，クラウドによる人出で文字誤りの校正を行っている．

一方図書館には古典，近代，現代の年代が異なるテキスト化されていない多種多様な膨大な資料があり，OCR の高精度化の研究が行われている．

本論では，テキストデータを画像に変換し，OCR を行い，正解と OCR の結果のパターンマッチングにより，その認識率をすべて自動で行えるテスト環境を構築した．これにより，Wikipedia から約 36 万タイトル，総文字数約 4.3 億個から，個々の文字の認識率と何の文字にどのくらいの確率で誤っているかの統計データを得た．

この OCR の誤り率と，事前学習済み BERT[1]を使って単語を MASK し，その単語を推測して得られた候補とを組み合わせることにより，OCR の誤りの修正のテストを行った．事前学習済み BERT は 3 種類でおこなったので，その結果を報告する．

本稿の構成は，次の通りである．第 2 節で，本論で使用している BERT 等の要素技術を関連研究として概説する．第 3 節では，大量のテキストから個々の文字認識率を測定するために構築したテスト環境を概説する．第 4 節では，第 3 節で構築したテスト環境において，画像の解像度・学習データ等による，認識率への影響のテスト結果を報告する．第 5 節で，事前学習済み BERT と，第 4 節で得た個々の文字の認識結果を組み合わせた，OCR の後処理による認識率の改善のテスト結果について報告する．第 6 節でまとめと今後の方向性について述べる

２．既存研究

本研究では，OCR の誤り訂正のために，自然言語処理での文脈による手法を用いているため，この分野での関連研究を概説する．

自然言語処理の分野では，単語をベクトルで表す研究は盛んに行われてきており，特に 2013 年に Mikolov らが考案した Word2Vec[2]により，語彙数より少ない次元

†1 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, the University of Tokyo
†2 国立情報学研究所，東京大学生産技術研究所
National Institute of Informatics/Institute of Industrial Science, the University of Tokyo

Embedding 数のベクトルを学習から得る方法が提案された．周辺の単語から対象単語を推測していく CBoW(Continuous Bag-of-Words)モデル，逆に単語からその周辺単語を予測する Skip-gram(Continuous Skip-gram)モデルである．これにより，似た文脈で出てくる単語同士の類似性をとらえることができるようになった．

2018 年 10 月には, Google の Jacob Devlin らにより, BERT (Bidirectional Encoder Representations from Transformers) が発表され，双方向 Transformer により，大規模コーパスから言語モデルを事前学習することにより，自然言語処理の 11 のタスク(GLUE では 8 タスク)で，SOTA(State of the Art)を達成した．教師ラベルの付いていない文章を使うことにより，大量のデータで学習を行っている．事前学習の手法として，一つは，Masked LM という手法で，文章の 15%を MASK トークンに置き換え, そのうちの 80%を MASK し，この MASK された単語を予測することにより, 前後の文脈を考慮する手法である．二つ目は，Next Sentence Prediction で, 2 つの文章が与えられたときに, 50%の確率で二つ目の文章を隣接していない別の文章に置き換え，これらが隣接文章かを判定する．本論で OCR の後処理として使用しているのが，この BERT の Masked LM での手法である．京都大学黒橋・河原研究室が，BERT の日本語に対応した事前学習モデル[3]を公開し，その後東北大学の乾・鈴木研究室でも事前学習済み BERT[4]を公開している．両者とも，この事前学習モデルの入力となるテキストは，日本語 Wikipedia 全文である．

３．大量のテキストから自動で認識率を測定するテスト環境の概要

大量のテキストデータから OCR を行い，認識率の算出を自動で行うテスト環境を構築した．下記の 4 個のコンポーネント(プログラム)で構成している．

- Wikipedia のダンプからタイトル毎のファイルを作成
- テキストを画像に変換
- OCR ソフトウェアによる文字認識
- 認識結果と入力テキストのパターンマッチングによる認識率の算出

以下，それぞれのコンポーネントを説明する．

3.1 Wikipedia のダンプからタイトル毎のファイルを作成

Wikipedia については, メタデータや本文テキストのダンプファイルが公式に提供されており，ダンプファイルから必要な情報を抽出するために広く使われているツール(wikiextractor)が既に存在するため，これらを利用して xml に変換することで，タイトル毎の個別のテキストファイルを作成した．これにより，大量のテキストデータが入手できた．

3.2 テキストを画像に変換

テキストから画像に変換するプログラムは 2 種類作成した．

1 つ目は，python の画像処理ライブラリ Pillow の ImageDraw 内の，画像にテキストを描画する text メソッドを使い，テキストを直接画像に変換する方法である．

2 つ目は， python の複数のライブラリを使用することにより，テキストを word 文書に変換し，更に pdf に変換し，画像に変換する方法であり，一連の処理を自動で行えるようにした． OCR は一般的には pdf から行うことが多いため，本稿では，性能評価のために 2 つ目の方法を採用した．

以上により, Wikipedia に含まれる大量のテキストデータをもとに, 例えば 10.5 ポイント, MS 明朝, 太字のように, フォントのサイズ，種類，属性を任意に指定可能な，OCR 検証画像データを作成するためのパイプラインを構築した．

3.3 OCR ソフトウェアによる文字認識

OCR ソフトウェアとして, オープンソースソフトウェアの Tesseract[5]を採用した. Tesseract は, 開発元が Google で, 日本語を含めた多くの言語の認識をサポートし，最新版では, LSTM モデルを利用した OCR エンジンが搭載されている．

3.4 認識結果と入力テキストのパターンマッチングによる認識率の算出

OCR の認識率を測定するためには，入力のテキストと OCR の結果のテキストの文字列の互いの位置を揃える必要がある．このためレーベンシュタイン距離を基準にパターンマッチングを行った．レーベンシュタイン距離は，二つの文字列が最小編集距離で表される距離である．挿入・削除・置換の操作によって，一方の文字列をもう一方の文字列に変える回数が最小になる手順として定義される．ただし，1 回の操作で 1 文字だけ操作する．このアルゴリズムをベースにすることにより，入力のそれぞれの文字と OCR の結果が対応できるので, 認識率を測定することができる．

本稿では，認識率として下記の 2 種類を定義した．レーベンシュタイン距離での文字数 S=一致の数 D=削除の数 I=挿入の数 R=置換の数とすると，

認識率_i = S/(S+D+I+R) = S/(入力文字数+I)

認識率_o = S/(S+I+R) = S/OCR の結果の文字数

個々の文字の認識率_i は，入力文字が，OCR による結果の認識された文字の集合より算出する．認識率_o は，逆に OCR の認識結果の文字に対応する, 入力文字の集合から算出する．

４．大量のテキストデータからの認識率の測定

第 3 節で構築したテスト環境を使って，2 通りの条件で OCR の文字認識のテストを行った．なお，パターンマッチングをする際，前処理として，

- テキストの空白と改行は削除

- テキストの半角文字は全角に変換

を行った.

1 つ目は，下記のテスト条件で行った.

- フォントは MS 明朝，フォントサイズは 10.5pt.
- OCR ソフトウェア Tesseract の version は 3 種類（およそ 4 カ月毎にリリースされている）
  v4: 1.0.20190314
  v5: 0.0-alpha.20191030
  v51. 0.0-alpha.20200223
- Tesseract の入力画像の解像度（pdf を画像に変換した時の解像度）200DPI , 300DPI, 400DPI の 3 種類
- Tesseract の 2 種類の学習データ
  fast: 標準の学習データ
  best: 認識精度が高い，処理速度が遅い学習データ
- Wikipedia タイトル数 200 個

以上の条件で行った結果が表 1 である.

2 つ目のテスト条件は，下記である.

- ＭＳゴシック 10.5,12,14pt で 200dpi,fast
- Wikipedia タイトル数 200 個

テスト結果は表 2 である.

表 1 入力解像度と学習データでの認識結果

test case	タイトルの数	認識率_i	入力の文字数 (S+R+D)	OCRの文字数 (S+R+i)	一致(S)	挿入(I)	削除(D)	置換(R)
_v4_200dpi_fast	200	0.98297	951852	953369	940184	4619	3102	8566
_v5_200dpi_fast	200	0.98297	951852	953369	940184	4619	3102	8566
_v51_200dpi_fast	200	0.98297	951852	953369	940184	4619	3102	8566
_v5_200dpi_best	200	0.98274	951852	957212	942239	6933	1573	8040
_v5_300dpi_fast	200	0.65935	951852	952643	663899	55040	54249	233704
_v51_300dpi_fast	200	0.65935	951852	952643	663899	55040	54249	233704
_v5_300dpi_best	200	0.64651	951852	957742	653078	58307	52417	246357
_v5_400dpi_fast	200	0.97005	951852	949471	929763	6620	9001	13088
_v51_400dpi_fast	200	0.97005	951852	949471	929763	6620	9001	13088
_v51_400dpi_best	200	0.97898	951852	958783	940286	8619	1688	9878

表 2 MS ゴシックでの認識結果

test case	タイトルの数	認識率_i	入力の文字数 (S+R+D)	OCRの文字数 (S+R+i)	一致(S)	挿入(I)	削除(D)	置換(R)
MSゴシック 10.5pt	200	0.97623	1049852	1051785	1031579	6844	4911	13362
MSゴシック 12pt	200	0.97623	1049852	1051785	1031579	6844	4911	13362
MSゴシック 14pt	200	0.97888	1049852	1051780	1034055	6512	4584	11213

表 3　約 36 万タイトルでの認識結果

test case	タイトルの数	認識率_i	入力の文字数 (S+R+D)	OCRの文字数 (S+R+i)	一致(S)	挿入(I)	削除(D)	置換(R)
_v5_200dpi_fast	358158	0.97792	429324059	430362993	422450960	2664135	1625201	5247898

4.1 大量のテキストからの認識率測定

以上の 2 つのテスト結果により, Tesseract のバージョン・フォント・フォントサイズによる影響は無かった．入力画像の解像度と学習データの結果は, 予想とは異なっていた．大量のテキストデータから認識率を測定するためのテスト条件は,

- MS 明朝 10.5pt で 200dpi,fast
- Wikipedia タイトル数約 36 万個

で行い，その結果のサマリーが表 3 である.

なお，表 3 において，挿入 I と削除 D が多い原因の一つとして，下記のようなバースト文字認識エラーが，ところどころで発生していた.

入力文：

2002 年（平成 14 年）2 月 3 日から 2003 年（平成 15 年）全 50 話が放映された東映制作の特撮テレビドラマ作品，および作中で主人公が変身するヒーローの名称である．（引用: Wikipedia の「仮面ライダー龍騎」より）

OCR の結果の文（一部空白と空行は削除している）:

2002 年「仮面ラ RRYUKI"(平成)1 月 19 日全 50 詰が放映された東映制までデレビ朝日ヒーローの名称である.

５．OCR の後処理での認識率の改善

この節では，前節で得られた個々の文字認識率と，BERT での MASK から得られた候補を組み合わせて，OCR の後処理での文字修正について述べる．後処理のアルゴリズムを，以下例をあげながら説明する.

S1.　後処理の文字の範囲を指定する．（文字の種類と個々の文字の認識率）

例: 0.90 から 0.91 の範囲の文字　人', 0.9057

S2.　OCR の認識結果の文から，該当の文字を含む単語を選ぶ.

例文: 言語は，人間が用いる意志伝達手段であり,

文字: 人 単語: 人間

S3.　大量の OCR 認識結果から得られた該当の文字の上位 5 個の候補を選び出す.

例:　人と OCR 認識した時の正解の文字と割合:
['人', 0.9057], [' ', 0.0706], ['入', 0.004],
['▲', 0.001], ['八', 0.0006]

他の文字を連結する: ['人間', 0.9057], ['　間', 0.0706],
['入間', 0.004], ['▲間', 0.001], ['八間', 0.0006]

S4.　文の中の単語を，MASK して BERT により Mask 部分を推測する．そして上位 100 個の候補を選び出す.

例: BERT の token : ['言語', 'は', '　', 'mask', 'が',

推測した候補: '人間', '個人', '社会', '人', '人々', '人類', '集団',

S5.　S3 で得られた候補を上から順に, 4 で該当している単語をさがす.

例: 人間が選ばれる

5.1 3 種類の事前学習済み BERT による評価

事前学習済み BERT として，京都大学の黒橋・河原研究室からは，単語ベースの 2 つを,

京大 1:WWM(Whole Word Masking)版(12 層)

京大 2: LARGE WWM 版(24 層)

東北大学の乾 鈴木研究室からは，文字ベースの 1 つ

東北 3: char-4k_WWM 版(12 層)

の 3 種類で,

MS 明朝 10.5pt 200dpi, fast

Wikipedia タイトル数 200 個

でテストを行った．なお，パターンマッチングの後，入力のテキストと OCR の結果のテキストを，文の単位で分割

し，レーベンシュタイン距離に基づいて，ペアの文を生成した．

テストでは，更に細かい条件を設定したので，以下その条件を列記する．

- ペアの文に対して 7 文字以下の短い文，384 文字以上の長い文は除外．
- 5 万文字を超えたペアの文は除外．

形態素解析を行う際の条件としては，

- 京都大学の事前学習済み BERT は，形態素解析エンジンとして Juman++であるが，MeCab[6]を使用した．
- 東北大学の BERT の時は，英数記号は半角に正規化した．

BERT で未知語となった token は，そのまま MASK で処理を行い 100 個の候補を得るが，文字の誤りの候補は，未知語に該当している文字列を用いた．BERT で"##"付きの sub-word に対しても，同様の処理を行った．

以上の条件で行ったテスト結果が表 4 である．表 4 では，OCR の後処理を行う文字は全種類（タイトル数約 36 万個の時 2689 種類）で，認識率_o は下限と上限の範囲を指定して行った．

表 4 の結果から，OCR の後処理後の認識率は，それぞれ，京大 1(12 層)が 0.962，京大 2(24 層)が 0.9606，東北大 4 は 0.7826 であった．京大 1 と京大 2 では，ほぼ同じ認識率であるので，BERT の層が多くなることによる影響は無かった．東北大 4 は，文字単位なので，後処理には不利であった．認識率 0-0.98 の範囲は，OCR の後処理による効果があったが，認識率 0.98-1.0 の範囲は，かえって認識率が下がる結果になってしまった．

5.2 文字種の絞り込み後の OCR の後処理

表 4 の結果より，OCR の後処理で効果のある文字と無い文字があることが判明した．従って，以下の方法で OCR の文字種を絞り込んだ．

S1. OCR の認識結果と BERT での後処理の結果により，各々の文字に対して，OCR の正解から後処理が正解(G2G)，正解から誤り(G2B)，誤りから正解(B2G)，誤りから誤り(B2B)の個数を算出する．

S2. 改善率(= B2G / (G2B + B2G))以上の文字を選ぶ．

S3. 改善数(= B2G - G2B)以上の文字を選ぶ．

Wikipedia のタイトル数を 20000 個に増やし，京大 1 で認識率_o の範囲を 0-0.98 の範囲で再度後処理の効果を確認した結果が表 5 である．表 5 の左側の部分が，文字種を全部で行った時の結果である．真ん中の部分が，前述の文字の絞り込み条件として，改善率の閾値を 0.75，改善数を 20 と設定し場合，文字種 519 個が選び出された時の結果である．右側の部分は，別の Wikipedia の 20000 個で，文字種 519 個で行った結果である．36 万タイトルの Wikipedia での OCR の認識全文字種は，2689 種であり，認識率_o の範囲 0-0.98 の間に出現した文字の数は，1486 種であったので，絞り込んだ文字種 519 種は，全文字種のおよそ 1/5 であった．

なお，表 5 で認識率_o の 0-0.2 の範囲の BERT での後処理で，改善文字数が多いのは，出現頻度の高い '］'と'＄'と'ゃ'の 3 文字の改善が多かったためである．'ゃ'は，認識率_i は 0.99539 で，認識率_o は 0.10692 であったので，'ゃ'は

表 4　3 種類での BERT の後処理の結果（タイトル数 200）

タイトル数約36万個					タイトル数200個 京大1			タイトル数200個 京大2			タイトル数200個 東北大3		
認識率加減	上限	文字の種類	文字数	全体からの比率	該当文字数	後処理の文字数	後処理の認識率	該当文字数	後処理の文字数	後処理の認識率	該当文字数	後処理の文字数	後処理の認識率
0	0.2	31	974465	0.002	2282	1816	0.6366	2282	1843	0.6294	2263	2159	0.5516
0.2	0.4	39	297415	0.001	562	452	0.7257	562	456	0.739	558	514	0.7374
0.4	0.6	79	485556	0.001	839	568	0.8398	839	612	0.8497	824	693	0.8442
0.6	0.8	182	2475777	0.006	4320	3188	0.9366	4320	3316	0.9415	4314	3666	0.743
0.8	0.9	271	5154337	0.012	9928	7963	0.9409	9928	8366	0.9429	9857	8607	0.867
0.9	0.92	107	6277275	0.015	13849	12037	0.9447	13849	12380	0.9482	13819	7820	0.8183
0.92	0.94	142	5542712	0.013	12448	9827	0.9724	12448	10411	0.9729	12422	6798	0.8883
0.94	0.96	211	8480936	0.02	15962	12642	0.9745	15962	13604	0.9794	15940	9282	0.8371
0.96	0.98	424	40157014	0.093	92035	76153	0.9765	92035	80600	0.9807	91905	50031	0.8309
0.98	0.99	466	92594220	0.215	203785	165684	0.9643	203785	167965	0.963	203454	119423	0.8265
0.99	1.1	731	267922887	0.623	581987	496309	0.9621	581987	502174	0.9581	581080	383696	0.7585
合計					937997	786639	0.9627	937997	801727	0.9606	936436	592689	0.7826

表 5　BERT の後処理の結果（タイトル数 20000）

タイトル数約36万個			タイトル数20000個 京大1 文字種全部				タイトル数20000個 京大1 文字種制限					タイトル数別の20000個 京大1 文字種制限				
認識率加減	上限	全体からの比率	該当文字数	認識率	後処理の文字数	後処理の認識率	該当文字数	認識率	後処理の文字数	改善率	後処理の認識率	該当文字数	認識率	後処理の文字数	改善率	後処理の認識率
0	0.2	0.002	138354	0.0604	111136	0.6394	137140	0.0597	110710	64333	0.6408	87806	0.0596	69453	41853	0.6621
0.2	0.4	0.001	31381	0.202	22948	0.7058	29816	0.196	22569	11822	0.7055	20956	0.2142	15800	8415	0.7379
0.4	0.6	0.001	59069	0.5122	38484	0.8317	44368	0.5281	27975	8375	0.9435	31229	0.4889	19305	6250	0.9341
0.6	0.8	0.006	301051	0.7535	214592	0.9095	115241	0.7353	78994	8115	0.9701	84903	0.7215	58304	6690	0.9665
0.8	0.9	0.012	655103	0.8691	528997	0.9335	332856	0.8715	265974	11906	0.9632	239299	0.8709	192166	9326	0.965
0.9	0.92	0.015	785230	0.9121	656584	0.9284	130862	0.9132	108149	2002	0.974	90415	0.9181	78357	1673	0.9761
0.92	0.94	0.013	705327	0.9336	549713	0.961	430149	0.9325	341839	10542	0.972	281932	0.9257	238074	8341	0.9721
0.94	0.96	0.02	1078762	0.9511	850643	0.969	361543	0.956	289595	3514	0.9903	247171	0.9535	209208	2859	0.9904
0.96	0.98	0.093	5304013	0.9719	4393600	0.9722	1128491	0.9722	960047	4740	0.9922	754324	0.9715	666415	4128	0.9931
合計			9058290	0.927	7366697	0.9559	2710466	0.8765	2205852	125349	0.9625	1838035	0.8744	1547082	89535	0.9659

OCR ではほとんど正しく認識されるが，他の文字が'や'と誤認識が多いことを示している．

5.3 OCR の後処理の結果のまとめ

表 5 の結果より，0-0.98 までの認識率の範囲での，906 万文字で改善した文字数が 125349 なので，認識率への改善は 0.014 であった．他の Wikipedia20000 個のタイトルでの結果は若干下がっていた．0.98-1.0 の範囲の文字の比率は表 4 の値により全体の 0.838 なので，全体の文字数は，認識率_o の 0 から 0.98 までの文字数のおよそ 5 倍とみなすことができる．従って，文字全部に対する OCR の改善は，およそ 0.2%であった．

以上により，表 3 の認識率 0.978 が後処理により 0.980 に改善したことになる．

６．おわりに

本稿での提案手法をまとめると

- 大量のテキストで OCR を行い個々の文字の認識率を算出する．
- 事前学習済み BERT の MASK で得られた候補と，各文字の認識率より得られた候補を組み合わせて，OCR の結果の文字を修正する．
- この時，効果のある文字を前もって絞り込んでおく．

である．

京都大学の事前学習済み BERT を使用して，文字種を約 1/5 に絞り込んで行った後処理のテストで，おおよそ 0.2%の認識率の向上になった．OCR のテストは，実際の画像のゆがみやよごれや傾きの補正を含めた総合的な検証を行われるが，本稿のようにテキストから直接画像に変換して，ノイズが無い状態でも，大量のデータで検証することは，OCR の基本特性を検証する上で有効な方法であることが確認できた．

今後は OCR エンジンの内部に踏み込んで，OCR の認識率の改善を検討したい．また，今後も視覚障害当事者の観点から，視覚障害者の情報障害の課題に取り組んでいこうと思っている．

参考文献

[1] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[2] Mikolov, Toma, Wen-tau Yih, and Geoffrey Zweig. "Linguistic regularities in continuous space word representations." Proceedings of the 2013 conference

[3] BERT 日本語 Pretrained モデル- KUROHASHI-KAWAHARA LAB http://nlp.ist.i.kyoto-u.ac.jp/index.php?BERT%E6%97%A5%E6%9C%AC%E8%AA%9E Pretrained%E3%83%A2%E3%83%87%E3%83%AB

[4] Pretrained Japanese BERT models released _ 日本語 BERT モデル公開 Tohoku NLP Lab _ 東北大学 乾・鈴木研究室 https://www.nlp.ecei.tohoku.ac.jp/news-release/3284/

[5]GitHub - tesseract-ocr_tesseract Tesseract Open Source OCR Engine (main repository) https://github.com/tesseract-ocr/tesseract

[6] MeCab: Yet Another Part-of-Speech and Morphological Analyzer https://taku910.github.io/mecab/