Part of Speech Tagging in Javanese using Support Vector Machine Method

Fa'iq Askhabi[1], Arie Ardiyanti Suryani[2], Moch. Arif Bijaksana[3]

Fakultas Informatika, Universitas Telkom, Bandung

[4]Divisi Digital Service PT Telekomunikasi Indonesia

[1]askhabifaiq@student.telkomuniversity.ac.id, [2]ardiyant@telkomuniversity.ac.id, [3]arifbijaksana@telkomuniversity.ac.id

Abstraksi— Part Of Speech (POS) Tagging untuk Bahasa Jawa menggunakan metode *Support Vector Machine* (SVM). Bahasa Jawa merupakan salah satu Bahasa daerah di Indonesia, terutama di pulau jawa bagian tengah sampai timur. Dalam penerapan metode SVM ini data yang digunakan diambil dari berita online dengan Bahasa Jawa. SVM sudah banyak digunakan untuk melakukan klasifikasi teks, namun untuk kasus POS Tagging masih sedikit dan khusus untuk Bahasa tertentu mungkin belum pernah ada yang menerapakannya. Sehingga kami ingin menerapkan metode SVM untuk kasus POS Tagging Bahasa Jawa. Dalam pengujian model yang sudah kami buat hasil terbaik yang kami dapatkan memiliki akurasi 77% dengan total jumlah label 20.

Keywords—Part of Speech, SVM, tagger, Bahasa Jawa.

Abstrac— Part of Speech (POS) Tagging for Javanese uses the Support Vector Machine (SVM) method. Javanese is one of the regional languages in Indonesia, especially in the central to eastern Java islands. In the application of this SVM method the data used is taken from online news in Javanese. SVM has been widely used to classify text, but in the case of POS Tagging is still small and especially for certain languages may have never been applied. So we want to apply the SVM method for the Java POS Tagging case. In testing the model we have made the best results we get have an accuracy of 77% with a total number of labels of 20.

Keywords—Part of Speech, SVM, tagger, Bahasa Jawa.

I. Pendahuluan

Part Of Speech (POS) Tag adalah pelabelan kata dalam kalimat (corpus) sesuai dengan jenis katanya[10]. POS Taging merupakan bagian dari Natural Language Processing (NLP) yang diterapkan dalam pengenalan suara (speech recognition), pencarian informasi (information retrieval), pengucapan teks (text to speech), pengolahan semantic (semantic processing), dan mesin penerjemah (machine translate) (Jurafsky and Martin, 2000).

Penelitian tentang POS tag dengan berbagai Bahasa sudah banyak dilakukan, metode yang digunakan juga berbeda dan memiliki akurasi yang tinggi. Beberapa metode yang sudah pernah diterapkan antara lain POS Tagging Bahasa Indonesia Dengan HMM dan Rule Based dengan akurasi 92.2% (Kathryn Widhiyanti dan Agus Harjoko, 2012). Kemudian ada juga POS Tagger Bahasa Odia Part of speech tagging in odia using support vector machine dengan akurasi 82% (Bishwa Ranjan Dasa, dkk., 2012).

Penelitian sebelumnya sudah pernah menerapkan POS Tagging untuk Bahasa Jawa namun berbeda metode. Paper tersebut adalah Pengaruh POS Tagging Berbasis aturan dan Distribusi Probabilitas Maximum Entropy untuk Bahasa Jawa krama dengan akurasi 97.76% (Hafiz Ridha Pramudita, Ema Utami, Armadyah Amborowati, 2016). Metode Rule Based ini merupakan metode yang menggunakan aturan Bahasa (grammar) untuk mendapatkan kelas kata pada suatu kata dalam suatu kalimat (Jurafsky, 2000).

Metode Rule Base ini memiliki 2 arsitektur, yaitu metode Rule Based yang menggunakan kamus untuk menandai kata dengan kelas kata (leksikon). Metode yang kedua adalah menggunakan disambiguation rule secara manual yang nantinya diproses menjadi satu kelas kata saja untuk setiap kata (Jurafsky, 2000). Namun referensi untuk rule based ini sangat sulit ditemukan karena harus berhubungan langsung dengan pakarnya.

Penelitian ini dilakukan untuk membuat POS Tagging dengan metode Support Vector Machine untuk Bahasa Jawa. Karena untuk penelitian mengenai POS Tagging Bahasa jawa menggunakan SVM belum pernah dilakukan. Kemudian jika melihat beberapa penelitian terkain POS Tagging yang menggunakan SVM akurasi yang dihasilkan sudah bagus sehingga penulis tertarik untuk mencobanya ke dalam teks Bahasa Jawa. Sehingga diharapkan hasil dari penelitian ini menghasilkan pelabelan yang akurat dengan memiliki akurasi yang tinggi.

II. Literature Survey

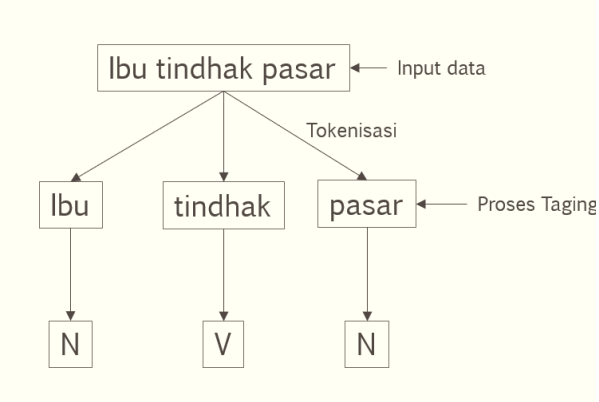
Pada penelitian yang dibuat oleh *Kathryn Widhiyanti dan Agus Harjoko* metode yang digunakan ada 2 yaitu menggabungkan antara *HMM* dan *Rule Based* pada tahun 2012. Dalam penelitiannya data yang digunakan dibagi menjadi dua, yaitu Data Training dan Data Uji. Untuk corpus yang digunakan didapatkan dan dimodifikasi dari beberapa penelitian yang sudah ada yaitu Corpus pertama (corpus 1) merupakan modifikasi corpus penelitian HMM Based POS (Wicaksono & Purwarianti,2010). Corpus yang kedua (corpus 2) adalah modifikasi corpus penelitian dengan metode CRF dan Maximum Entropy (Pisceldo, Andriani dan Manurung).

Untuk teks masukan yang tidak sama dengan teks dalam corpus, pada pengujian terhadap corpus I kedua metode memiliki hasil akurasi yang sama dan akurasi tertinggi pada percobaan ini adalah 80,48%. Sedangkan pada pengujian terhadap corpus 2 diperoleh hasil bahwa akurasi dengan HMM lebih tinggi dibanding dengan menggunakan HMM dan Rule Based. Pada penggunaan HMM saja diperoleh akurasi tertinggi 92,91% sedangkan pada penggunaan HMM dan Rule Based akurasi tertinggi yang diperoleh adalah 92,2%.

Kemudian penelitian untuk POS Tagging Bahasa Jawa menggunakan metode *Berbasis aturan dan Distribusi Probabilitas Maximum Entropy pada tahun 2016 oleh* Hafiz Ridha Pramudita, dkk. *Distribusi Probabilitas Maximum Entropy* salah satu pendekatan statistik dalam penandaan kalimat yang terus mengalami pengembangan. Data didapatkan dari jurnal ilmiah dari Universitas Negeri Yogyakarta dan sudah memiliki anotasi manual untuk data training dan hasil pengukuran akurasi. Pada penelitian ini data didapatkan dibagi menjadi dua macam, yaitu data train dan data test. Tagset dan kamus yang digunakan bersumber pada buku kosa kata jawa dengan jumlah kata 10.000. Dari semua pengujian dengan metode tersebut didapatkan akurasi pelabelan kata sebesar 97.67%.

III. Metode

Perancangan sistem pada penelitian ini secara umum hampir sama seperti sistem pada penelitian lain tentang POS Tagging. Secara umum POS Tagging akan memberikan label untuk setiap kata yang diinputkan, sebagai contoh seperti pada gambar 1. Namun pada metode SVM harus dilakukan vektorisasi karena SVM adalah classifier yang menggunakan vector untuk memprediksi input data.



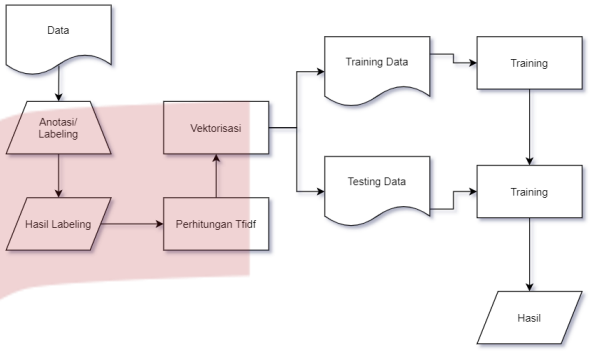
Gambar 1 Gambaran Umum POS Tagging

Pada penelitian ini jumlah label atau kelas kata yang digunakan adalah 19. Label yang akan digunakan bisa dilihat pada Tabel 1, namun ada beberapa label yang dilakukan penyesuaian, misalkan nilai mata uang akan dimasukkan kedalam label SYM karena itu tidak bisa dibuat satu per satu.

Tabel 1 Tag set

No.	Tag Set	Ket. Tag	Contoh
1	N	Noun	Indah, Ayam
2	V	Verb	Tindhak, mangan
3	ADJ	Adjective	Apik, Ayu
4	ADV	Adverb	Mangkih, Bablas
5	KNJ	Conjuction	Lamun, Lan
6	PRP	Preposition	Marang
7	KH	Khusus	Banget
8	SO	Subordibator	Nalika
9	EM	Emotif	Eh, Aduh
10	PR	Pronomial	Niki, Kulo, Kuwi
11	SYM	Symbol	%, &, #, Rp
12	({[	Opening Parentheis	(, {, [
13	)}]	Closing Parentheis	), },]
14	,	Comma	,
15	‘ “	Quotation	‘, “
16	.,?,!	Sentence Termonator	? !
17	--,-	Dash	- --
18	:	Colon	:
19	;	Semicolon	;

Kemudian sistem yang kami rancang untuk melakukan POS Tagging menggunakan SVM bisa dilihat pada gambar 2.



Gambar 2 Rancangan sistem

a. Labeling

Semua data yang sudah didapatkan kemudian dibentuk menjadi corpus dan diberi label per kata. Sehingga data ini nantinya akan digunakan sebagai acuan untuk proses training dan juga testing. Dalam melakukan proses ini, label yang digunakan mengacu pada Tag Set Bahasa Jawa.

b. Pembobotan kata dan vektorisasi

Kata yang sudah diberi label kemudian dilakukan pembobotan menggunakan perhitungan Tfidf. Dengan Tfidf hasilnya akan digunakan untuk pembentukan vector agar bisa di proses didalam SVM. Dalam kasus ini tfidf yang digunakan melakukan pengecekan per character sehingga mampu memprediksi kata dengan baik, contoh hasil dari Tfidf bisa dilihat pada Tabel 2.

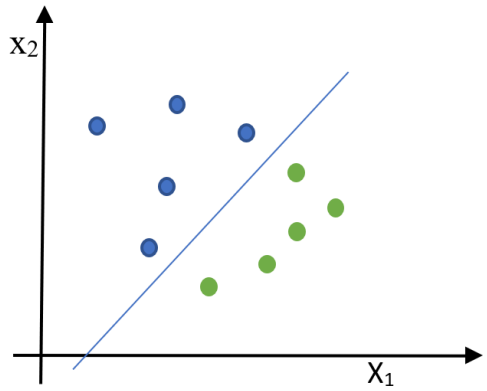
Tabel 2 Contoh Hasil Tfidf

Kata	Hasil Pembobotan
r	0.159
e	0.243
g	0.467
i	0.150
re	0.439
eg	0.220
gi	0.135
reg	0.356
egi	0.535
regi	0.233

c. Support Vector Machine

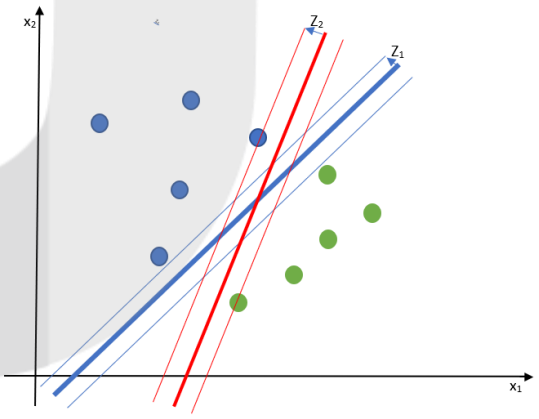
SVM merupakan sebuah metode klasifikasi yang memisahkan data dengan menggunakan hyperplane. Hyperlane adalah sebuah titik tertinggi yang memisahkan beberapa kelas untuk klasifikasi SVM. SVM dapat digunakan untuk menghasilkan satu atau beberapa hyperplane pemisah sehingga data terpisah menjadi beberapa segmen dan setiap segmen hanya berisi satu kelas data. Teknik SVM secara umum sangat berguna untuk data yang tidak diketahui distribusinya.

Fungsi SVM dalam NLP yaitu sebagai classifier dalam menghasilkan label kata sesuai jenisnya. Pada kasus ini SVM akan digunakan untuk memisahkan kelas kata dengan menentukan hyperlane yang terbaik untuk setiap kelas atau label. Cara untuk menemukan hyperlane terbaik adalah garis pemisah yang memiliki margin paling besar diantara kelas yang dipisahkan. Maka dapat dituliskan W.X + b = 0. Dimana W adalah bobot dari sebuah vector, W = {w_1,w_2,w_3,…,w_n}; dengan n adalah jumlah dari atribut yang dipakai dan b adalah bias.



Gambar 3 SVM memisakan kelas secara Linear

Dalam SVM untuk mencari hyperlane terbaik adalah dengan cara mengukur margin dari masing-masing hyperlane yang ada dan dicari titik maksimal dari margin tersebut. Margin merupakan jarak antara hyperlane dengan node terdekat dari masing-masing kelas. Pada gambar 4 dibawah adalah ilustrasi, asumsikan bahwa margin z2 memiliki nilai yang lebih besar dibanding dengan margin z1. Maka garis merah merupakan hyperlane terbaik untuk kasus tersebut.



Gambar 4 Menentukan hyperlane terbaik

Pada ruang cartesius 2D, persamaan garis $ax + by + c = 0$ tentunya sudah sangat familiar. Kita akan membuat persamaan yang lebih umum dari persamaan tersebut yang dapat mencakup ruang berdimensi banyak. Pertama-tama, kita ubah notasi variabel dan konstanta dari persamaan garis tersebut: x menjadi x_1, y

menjadi x_2, a menjadi w_1, b menjadi w_2, sehingga menjadi :

$w_1x_1 + w_2x_2 + c = 0$

Misalkan ada sebuah dimensi di mana $d > 1$, maka persamaannya menjadi

$w1x1+. . . +wdxd + c = 0$

$\rightarrow \sum_j^d w_j x_j + c = 0$ (1)

cara lain menuliskan persamaan di atas adalah dengan notasi vektor. Kita nyatakan variabel-variabel w_i dan x_i dengan vektor $X = [x1, \dots , w_d]^T$ dan $W = [w_1, \dots , 2_d]^T$. Dengan demikian, persamaan (1) dapat ditulis menjadi

$\langle w, x\rangle + c = 0$ (2),

dimana $w, x \in \mathbb{R}^T$ dan $\langle w, x\rangle = w^Tx$ (dot product).

Kemudian tahapan dalam SVM diantaranya adalah :

a. Training

Dalam melakukan training SVM akan menghitung persamaan terhadap setiap token yang diinputkan, Data training diambil dari corpus yang sudah dibuat dengan jumlah 80% dari total data yang ada. Kemudian dilakukan optimasi parameter agar mendapatkan nilai margin yang paling baik dengan cara Cross Validation dengan 10 Fold.

Dari data yang ada, akan di split menjadi 10 Fold secara random untuk melakukan pengujian parameter. Setiap 1 kali pengujian dilakukan sebanyak 5 kali sehingga waktu yang dibutuhkan cukup lama namun memiliki hasil yang lebih bagus.

b. Testing

Model dari hasil training kemudian dipakai untuk melakukan testing. Dalam proses testing parameter yang digunakan sudah paling optimal sehingga hasil yang didapatkan sudah cukup baik.

IV. Pengujian Dan Hasil

a. Jumlah data

Data yang digunakan untuk melakukan testing yaitu 20% dari total data yang ada di corpus. Pemilihin besar data ini sudah dilakukan pengujian beberapa kali dan mendapatkan pembagian paling optimal dengan 20% data test. Berikut hasil dari pengujian untuk penentuan data :

Tabel 3 Pembagian data

No.	Data Train	Data Tes	Akurasi
1	50 %	50 %	74 %
2	60 %	40 %	76 %
3	70 %	30 %	77 %
4	80 %	20 %	77 %

b. Pembobotan kata

Hal pertama yang dilakukan adalah memberikan bobot untuk setiap token. Token yang digunakan adalah data dengan jumlah 3000 kata yang diambil dari korpus. Pemberian bobot ini menggunakan perhitungan TF-IDF. Konsep TF-IDF sendiri adalah mencari token (term) dengan kemunculan paling banyak sehingga ini mampu membantu kinerja SVM agar lebih efektif. Berikut adalah contoh hasil perhitungan TF-IDF :

1	2	3
(0,	354)	0.20470525889561422
(0,	944)	0.3470926343694299
(0,	6565)	0.366232688848692
(0,	6566)	0.5113632745332957
(0,	6622)	0.6230472032656001
(0,	7580)	0.23198245995575836
(1,	2145)	0.06782165227659556
(1,	2806)	0.12437555310983853
(1,	2812)	0.1611747752493676
(1,	2815)	0.17727663330998022
(1,	2816)	0.17727663330998022

Gambar 5 Hasil hitung TF-IDF

Berikut adalah cara untuk melakukan perhitungan TF-IDF :

kulo wis mangan (d1)

ibu tindak pasar (d2)

bapak lagi gerah (d3)

Dari data tersebut asumsikan itu adalah 3 dokumen (D), sehingga jumlah dokumen total = 3. Kemudian dihitung untuk kemunculan (term) kata kulo muncul sebanyak 1 kali dalam dokumen 1 sehingga tf = 1. Setelah itu hitung bobot dengan rumus TF-IDF :

$$w_{ij} = tf_{ij}\, x \log(D/df_j) + 1$$

$$w_{ij} = 1\, x\, (\log(3/1) + 1)$$

$$w_{ij} = 1\, x\, (0.477 + 1)$$

$$w_{ij} = 2.954$$

Dimana w = bobot, tf_{ij} = jumlah kemunculan dalam semua dokumen, D = total dokumen yang diujikan, dan df_j = total dokumen yang mengandung term tersebut. Dari perhitungan diatas didapatkan hasil 2.954. setelah dihitung bobot untuk masing masing kata akan dilakukan vektorisasi menggunakan library *TfidfVectorizer* dari Scikit-Learn.

c. Cross Validation

Dalam pengujian ini kami melakukan Cross Validation untuk mengevaluasi model yang sudah dibuat. Dengan jumlah Fold 10 maka data displit secara random menjadi 10 bagian. Dengan split tersebut 9 bagian akan digunakan sebagai data latih dan 1 bagian lainnya untuk data uji. Pada tahapan ini parameter yang diuji adalah nilai besar margin dengan param yang diberikan 5 nilai berbeda. Untuk hasil pengujian bisa dilihat pada tabel 4.

Tabel 4 Hasil pengujian parameter C

Nilai C	Akurasi
0.1	63 %
0.5	74 %
1	76 %
1.5	77 %
2	77 %
3	77 %
4	77 %
5	77 %
10	77 %
100	77 %

Kami juga melakukan pengujian dengan nilai parameter lebih besar namun akurasi yang didapatkan tidak terlalu berbeda jauh. Sehingga dari pengujian tersebut nilai parameter terbaik adalah antara 1.5 – 2.

Parameter terbaik tersebut kemudian digunakan untuk melakukan testing dengan jumlah data sebesar 20%. Dengan begitu didapatkan hasil seperti pada tabel 5.

Tabel 5 Hasil Pengujian Tag set

No.	Tag	Precision	Recall	F1-score	Support
1	"	1.00	1.00	1.00	1
2	({[	1.00	1.00	1.00	5
3	)}]	1.00	1.00	1.00	5
4	,	1.00	1.00	1.00	37
5	-- , -	1.00	1.00	1.00	2
6	.,?,!	1.00	1.00	1.00	27
7	:	1.00	1.00	1.00	2
8	ADJ	0.40	0.67	0.50	15
9	ADV	0.73	0.69	0.71	32
10	KH	0.75	0.75	0.75	8
11	KNJ	0.56	0.57	0.56	35

12	N	0.91	0.80	0.85	284
13	NUM	0.82	0.94	0.87	33
14	PR	0.44	0.44	0.44	16
15	PRP	0.53	0.56	0.55	41
16	SO	0	0	0	0
17	SYM	0.67	0.67	0.67	3
18	V	0.56	0.74	0.63	61

Kemudian alam melakukan proses tokenisasi masih belum sebaik yang diharapkan karena dalam proses tokenisasi masih belum bisa melakukan pemisahan antara kata dan tanda baca yang mengikuti. Sehingga hasil saat melakukan pelabelan masih mendeteksi sebuah token tersebut sebagai tanda baca atau kata benda. Misalkan dalam input kata terdapat sebuah tanda baca, maka label akan menjadi 2 kemungkinan antara kelas untuk kata tersebut atau tanda baca.

Input : Ibu tindak pasar.

Pada input kata terdapat tanda titik di akhir kalimat. Program akan memberi label antara titik ataupun N karena tanda baca tersebut belum bisa di pisahkan saat proses tokenisasi.

d. Kernel

Kami juga melakukan pengujian Model SVM dengan kernel yang berbeda, yaitu kernel RBF dimana cara kerjanya berbeda dengan linear. RBF akan melakukan penentuan hyperplane secara dinamis artinya tidak linear. Namun pada RBF tidak bisa memberikan hasil yang baik jika nilai gamma yang digunakan berbeda. Untuk kernel Linear selalu memberikan hasil yang baik untuk semua nilai gamma.

No	**Kernel**	**Gamma**	**Akurasi**
1	RBF	1/N_Features	41 %
		1/(N_features* var X)	74 %
2	Linear	1/N_Features	76 %
		1/(N_features* var X)	76 %

V. Kesimpulan

Penelitian ini berfokus pada penerapan SVM untuk POS Tagging Bahasa Jawa dan mengevaluasi seberapa baik SVM untuk POS Tagging Bahasa Jawa. Kami melakukan pengujian dengan beberapa tahapan, Cross Validation untuk optimasi parameter, kemudian melakukan testing dengan mencoba semua kemungkinan parameter. Selain itu kami juga melakukan percobaan terhadap kernel yang berbeda yaituu Linear dan RBF, untuk hasil yang didapatkan Liner selalu mendapatkan akurasi yang baik walaupun nilai gamma yang berbeda. Sehingga untuk melakukan Tagging lebih baik menggunakan kernel Linear. Dan hasil akurasi untuk kasus ini adalah 77%.

References

[1] Abdiansah, "Support Vector Machines: Penjelasan Matematis dan Intuitif," 05 June 2015. [Online]. Available: https://ghifar.wordpress.com/2015/06/05/support-vector-machines-penjelasan-matematis-dan-intuitif/. [Accessed 07 November 2019].

[2] S. AULIA, S. HADIYOSO and D. N. RAMADAN, "Analisis Perbandingan KNN dengan SVM untuk Klasifikasi Penyakit Diabetes Retinopati berdasarkan Citra Eksudat dan Mikroaneurisma," vol. III, no. SVM and KNN, 2015.

[3] R. B. Das, S. Sahoo, C. S. Panda and S. Patnaik, "Part of Speech Tagging in Odia Using Support Vector Machine," in *Intelligent Computing, Communication & Convergence*, Bhubaneswar, 2015.

[4] D. Jurafsky and J. H. Martin, "Speech and Language Processing," vol. III, pp. 151-152, 2019.

[5] M. I. MUBAROK, "SUPPORT VECTOR MACHINE IN R (CLASSIFICATION)," 8 August 2018. [Online]. Available: https://muhammadilhammubarok.wordpress.com/2018/08/08/support-vector-machine-in-r-classification/. [Accessed 06 November 2019].

[6] A. Nietzio, "Support vector Machines for Part-of-Speech Tagging".

[7] H. R. Pramudita, E. Utami and A. Amborowati, "Pengaruh Part of Speech Tagging Berbasis Aturan dan Distribusi Probabilitas Maximum Entropy untuk Bahasa Jawa Krama," no. Rule Based, 2016.

[8] K. Widhiyantil and A. Harjoko2, "POS Tagging Bahasa Indonesia," vol. VIII, no. POS Tagging, 2012.

[9] Wikipedia, "Bahasa Jawa," 12 November 2019. [Online]. Available: https://id.wikipedia.org/wiki/Bahasa_Jawa. [Accessed 10 October 2019].

[10] Wikipedia, "Part-of-speech tagging," 11 September 2019. [Online]. Available: https://en.wikipedia.org/wiki/Part-of-speech_tagging. [Accessed 10 October 2019].