Application of Logistic Regression with Part-Of-The-Speech Tagging For Multi-Class Text Classification

Tomas Pranckevičius
Vilnius University, Institute of Mathematics and Informatics
Vilnius, Lithuania
tomas.pranckevicius@mii.vu.lt

Virginijus Marcinkevičius
Vilnius University, Institute of Mathematics and Informatics
Vilnius, Lithuania
virginijus.marcinkevicius@mii.vu.lt

Abstract — **Today, computing environment provides the possibility of carrying out various data-intensive natural language processing tasks. Language tokenization methods applied for multi-class text classification are recently investigated by many data scientists. The authors of this paper investigate Logistic Regression method by evaluating classification accuracy which correlates on the size of the training data, POS and number of *n*-grams. Logistic Regression method is implemented in Apache Spark, the in-memory intensive computing platform. Experimental results have shown that applied multi-class classification method for Amazon product-review data using POS features has higher classification accuracy.**

Keywords — tokenization, natural language processing; part-of-the-speech; multi-class classification; Logistic Regression

I. Introduction

Classification methods are unique data-processing features of machine learning [1]. The aim of classification is to identify and assign predefined class to a selected object, when the training set of object with class labels is given. Today, machine learning algorithms are supported by large data processing frameworks and includes usually special libraries. One of the earliest comparative works on text classification using supervised machine learning methods revealed that Support Vector Machine is top-notch classifier, compared to Decision Tree or Naïve Bayes [2]. Dumais et al. [3] also demonstrated the superiority of Support Vector Machine over Decision Tree and Naïve Bayes. Later Support Vector Machine method was chosen by many researchers even without any consideration and became the most popular technique for classifying the texts not only for English, but for many other languages. This paper focus only on Logistic Regression (LR) classification method, because it is less investigated, but it is still applied in practical tasks. LR method is realized for multi-class text classification with Apache Spark, the in-memory intensive computing platform.

To run text classification, in the first step data pre-processing stages text corpus must be created. A corpus can be defined as a collection of machine-readable authentic texts that is sampled to be representative of a natural language structures [4]. Corpus is important for NLP in terms of research including linguistic investigations. Corpus provides a bottom line for development of NLP systems and contributes in development of computational linguistics, annotation [5], Part-of-the-speech (POS) tagging, syntactic parsing, semantic tagging, alignment of parallel corpora etc. [6]. Such techniques in general can be named as tokenization. Tokenization is a method and part of NLP stages, which segments the stream of words or characters (units) into tokens (characters, numbers). Standard tokenization is simply just separating words at white space or punctuation [7] can be applied to transform a plain text into the acceptable input format also known as corpus [8]. Advanced tokenization methods include more properties and options that could be specified per each natural processing method. As example, using apostrophes, word segmentation, phrases, part of the speech. Tokenization is mostly known as very basic and general stage in natural language processing, but it must be considered as one of the most important stages, because failures in this stage will be continuously replicated to the next NLP steps and will cause further issues on the expected results [9].

This paper is organized as follows: section 2 presents the features of natural language processing; section 3 presents the experimentation and results; section 4 presents conclusions.

II. Featues of Natural Language Processing

There are many tokenisation and language processing techniques and methods, but in this paper, authors investigate one of tokenisation method – POS tagging and calculates classification accuracy which correlates on the size of the training data, POS and number of n-grams. In this section, some more modern features of natural language processing are presented.

A. Segmentation

There are many types of text segmentation, such as topic segmentation, sentence segmentation, word segmentation (splitting), paragraph segmentation, discourse segmentation. The main idea of segmentation is to define and find the boundaries in the text that could define segments [10]. Sentence segmentation is the problem of dividing a string of written language into its component sentences and word segmentation when sentences is having any white spaces

978-1-5090-4473-3/16/$31.00 ©2016 IEEE

between the words [7]. In English and some other languages, using punctuation or whitespace is probably the most popular way that allow to brake the sentences in to the words, separating each single word with punctuation or white space. However, this problem is not trivial due to the usage of the full stop character, which may or may not also terminate a sentence. When processing plain text, tables of abbreviations that contain periods can help prevent incorrect assignment of sentence boundaries.

B. *Part-of-speech Tagging*

Part of speech tagging (POS) marks the words in a text with special labels corresponding to the part-of-speech of the word in this context. Simply POS identify words as nouns, verbs, adjectives, adverbs, etc. POS tagging is thought of to be a crude kind of word sense disambiguation [11]. Tagger marks the words and assign the labels which corresponds to the POS of the word in that context. The rules might be applied by two approaches: statistical or rule-based and trained using corpora manually labeled (i.e. single noun (N), plural noun (NNS), verb (VB), verb, past tense (VBD)).

The text subcategorization, special extraction of nouns and verbs can become strong indication for accurate sentiment analysis when applying multi-class classification algorithms in large scale text messages analysis.

There are existing comparison studies, that conclude the effectiveness of adjectives, verbs, and adverbs, where subcategorization usually makes reasonable impact [12], [13] [14], [15].

C. *Stemming and Lemmatization*

Stemming is a process for reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. Its main use is as part of a term normalization process that is usually done when setting up Information Re-trivial systems. One of mostly known - the Porter stemming algorithm (or 'Porter stemmer') is a process for removing the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalisation process that is usually done when setting up Information Retrieval systems. Besides Porter Stemmer, there are more stemming methods available that implements reduction of the words to their root forms also knowns as word stem (lemma), such as Krovetz Stemmer, Lovins Stemmer, Paice Stemmer. In the experiment the Porter Stemming algorithm is applied [16].

D. *More Tokenization Methods*

There are more tokenization methods available and summary descriptions of them are presented following:

- *Named entity recognision:* Name entity recognition contains all the named entities, which are phrases that contain the names of persons, organizations, locations, times and quantities [17].
- *Apostrophes:* Used for possession and contractions (aren't, don't, can't).
- *Specific tokens:* applied to identify contact related information such as phone numbers, emails, addresses, invoice payment days, bank account or VAT number.
- Hyphens: considering that white space are not always required to every case, such as Mercedes-Benz, San Francisco-Los Angeles.
- *Normalization*: typically, normalization is used to deal with accents and diacritics that is related to the regions language is used. It also reduces all letters to lower case.
- *Equivalence classes:* creates relations between two or more normalized tokens, such as chair and furniture.
- *Phrase:* identify phrases as a small group of words that creates a meaningful unit. There are different types of phrases: noun (*vase of roses, book about*), verb (*had been living, will be going*), adjective (*very interesting*), adverbial (*very slowly*), prepositional (*near the sea*).
- *Syntactic Parsing:* recognize the sentence and its grammatically correctness by assigning the grammar function to each of word in the corpus.
- *Chunking*: implements word and sentence segmentation and creates labels on multi-token sequences. It presents word-level tokenization and POS tagging. The large boxes present higher-level of chunking. Chunking usually makes selections on a subset of the tokens [18].

III. Experimentation and Results

Amazon customers' review data for Apps for Android are selected for the investigation [19]. The total number of records is given by $n = 2638274$. The customer review fields include: review text – a written customer review about the product; overall – a rating given by the customer for the product (ratings from 1 to 5 are used in this research); helpful – presents user feedback about the quality and helpfulness of the review; summary – gives a short version of the customer review or subject matter. Only overall and review text data were used in the experiments.

{"reviewerID": "AUI0O345FBETYH6", "asin": "2845NNDFV5T", "reviewerName": "Customer ABC", "helpful": [0, 0], "reviewText": "Happy to finally find this application on the Android market. My brother has it on her iPhone and he loves it! Happy to see more apps like this!", "overall": 5.0, "summary": "Super app!!!", "unixReviewTime": 13014562800, "reviewTime": "01 17, 2014"}

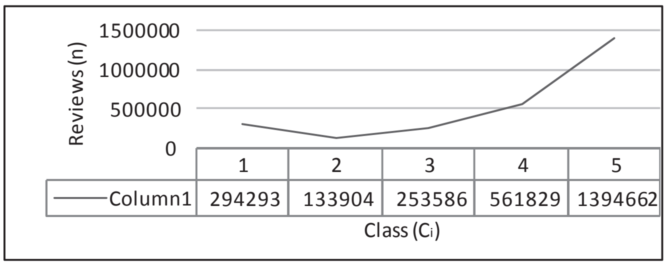
The data consist of different customer reviews given by $D = \{d_1, d_2, d_3, \ldots, d_n\}$, where n is the total number of reviews. These reviews are categorised by different customers, with each category assigned to the review with a rating value of $D = \{C_1, C_2, C_3, \ldots, C_5\}$, where C_i (i – class index), m is total number of classes ($m = 5$) is considered as a class label or class. The data class distribution C_i in the data set is presented in Fig. 1. To improve our model, it was decided to analyse an equality distributed data per class and use the measuring the skewness of data method [20], so that every class would collect an equal number of customer product-review records. Composition of data set for training and testing is distributed – 90% for training and 10% for testing (Fig. 2).

Very often there might be considered that reviews with 4–5 stars have very positive meaning, and all reviews below 2 stars – negative meaning. The review meaning is presented in Table 1.

TABLE I. REVIEW MEANING

Rating	Meaning
1 star	I hate it
2 stars	I don’t like it.
3 stars	It’s okay.
4 stars	I like it.
5 stars	I love it.

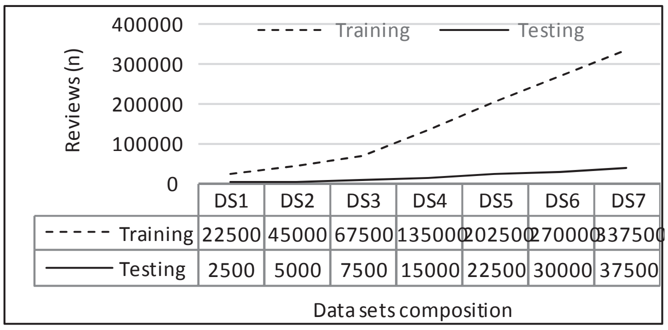
The data is selected only by required and related data fields to process the data and optimise memory usage. It is carried out by: selecting only *overall*, *review* fields and taking a defined number of reviews for each class.



	1	2	3	4	5
Column1	294293	133904	253586	561829	1394662

Fig. 1. Customer review distribution by classes.

Seven different sizes of data set, DS1 (25000), DS2 (50000), DS3 (75000), DS4 (150000), DS5 (225000), DS6 (300000), DS7 (375000) were used in experiments.



	DS1	DS2	DS3	DS4	DS5	DS6	DS7
Training	22500	45000	67500	135000	202500	270000	337500
Testing	2500	5000	7500	15000	22500	30000	37500

Fig. 2. Composition of data sets for training and testing.

A total of 10% of the reviews from each data set were used for testing.

A. Cloud Computing Infrastructure

Data-processing cluster infrastructure have been used during the experimentation: master with 2 vCPU (virtual central processing unit) and 26 GB of memory and two workers with 2 vCPU, with 13 GB of memory on each and 3 same level clusters have been used to process the tasks faster. The infrastructure was created at Google Cloud Platform. Experiments were performed using Apache Spark v1.6.0, Python v2.7.6 and NLTK v3.0.0. Apache Spark is an intensive in-memory computing platform designed to be one of the fastest available and very general-purpose in terms of running different kinds of computing task [21].

Fig. 3 presents the Apache Spark data processing cluster infrastructure model including Machine Learning library *spark.mllib* that was installed on Google Cloud platform.

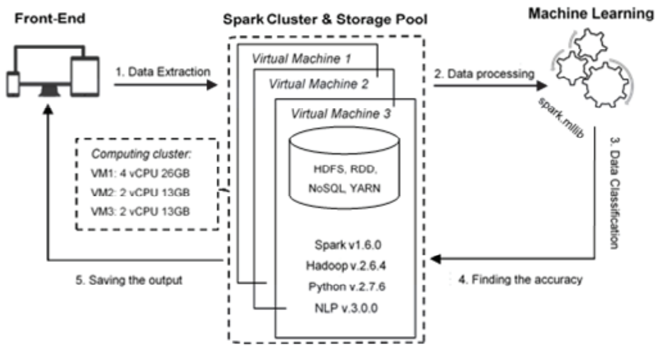


Fig. 3. Apache Spark data-processing cluster infrastructure.

B. Data Processing Stages

LR classification method in data processing model was used. This model (Fig. 4) is a modified version of that presented by Seddon [22]. In this section, unified data processing process is defined. The data pre-processing is the first and initial stage that must be considered at the beginning of engineering of the data analytics solution. Data pre-processing contains data extraction, data cleaning and data formatting. The main goal of this stage is to prepare data for classification step. The data processing workflow steps are following:

Step 1: Reading dataset. The main goal of this stage is to select only required and related data fields to process the data and optimize memory usage. This stage was carried out by:

- From input dataset are taken only Label and Text_Review fields.

Step 2: Pre-processing Text_Review (Fig. 4). The main goal of this stage is to prepare corpus for classification. This stage was carried out by:

- Applying word segmentation on each single word by punctuation or white space.
- Removing stop words and punctuation words.

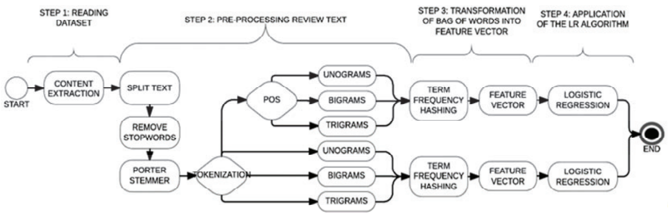


Fig. 4. Data processing workflow

- Stemming and reducing inflectional forms to a common base form stemma.
- Applying POS tagging: marks the words in a text with special labels corresponding to the POS of the word in this context. Simply POS identify words as nouns, verbs, adjectives, adverbs.
- *n*-gram as the *sequence of written words of length* n *is applied "unigram" for unigram* $p(W_i)$ *"bigram" for bigram* $p(W_i|W_i{-}1)$*, and "trigram"*

$p(W_i|W_i - 2)$, where W_i is the word in the sentence and

p is the probability.

Step 3: Transformation of Bag of words into Feature vector:

- Using HashingTF from Machine Learning Library of Apache Spark. These words are imported to a specially created hashing term-frequency vectorizer, which counts the frequency in the set and assigns a unique numerical value for the classification stage and weights need to be assigned to each word.
- Feature vector transforms text to the numerical meaning represented by the integers format and are ready to be process to the classification step.
- Dividing the corpus into two group, roughly 90% in training and 10% in test.

Step 4: Application of the LR algorithm:

- Using LR classification method to train and test the corpus set.
- Running cross-validating with 10 folds (cycles to run the model).
- Calculating the average classification accuracy for the test corpus.

C. *Results*

Fig. 5 and Fig. 6 consists of LR algorithm classification accuracy dependency on the number of *n*-gram, POS word features and size of product-review sets. The average accuracy formula for multi-class classification can be presented following [23]:

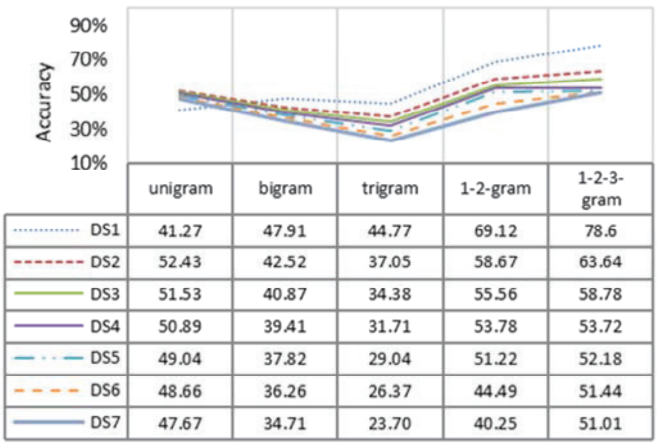
$$Accuracy = \frac{\sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i}}{l} \times 100\%, \qquad (1)$$

where tp_i are true positive, fp_i is the false positive, fn_i is the false negative, and tn_i – true negative counts respectively, l is the number of classes and multiplied by 100 for percentage value. Classification accuracy is counted of how many actual labels are equal to predicted label divided by total corpus in test, i.e. counting of (*actual class* = *predicted class*). The classification accuracy was collected after every computing tasks.

Fig. 5 presents that the classification accuracy of the LR classifier without POS. In comparison to unigram (min 41.27%, max 47.67%), classification accuracy decreases with bigram and trigram models and increases when a combination of uni/bi/tri-gram models are applied (min 40.25%, max 78.6%). Fig. 6 presents that the classification accuracy of the LR classifier with POS tagging. In comparison to unigram results (min 41.47%, max 40.06%), classification accuracy decreases with bigram and trigram models and increases when a combination of uni/bi/tri-gram models are applied (min 50.42%, max 78.29%).

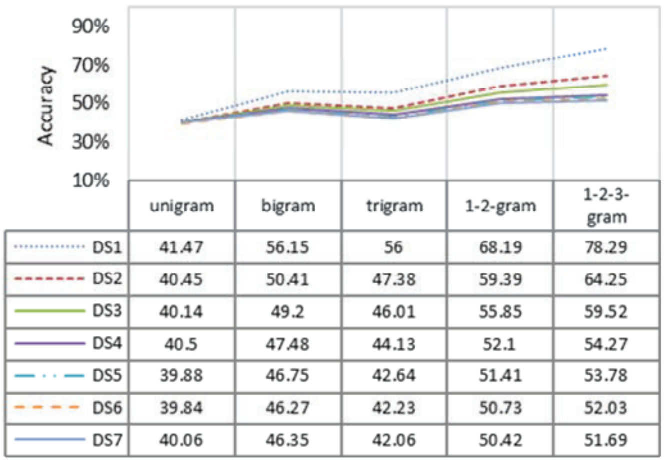
Comparing unigram results, classification accuracy overall is decreasing when increasing the size of the data set and combination of uni/bi/tri-gram models increase the average accuracy, but decreases when bigram and trigram models are applied.

Fig. 7 presents the average classification accuracy of the LR classifier without POS and with POS. Fig. 7 results have shown that applied LR multi-class classification method for product-review data using POS features has higher classification accuracy and using POS features has in average (3.31%) higher classification accuracy. Summary results of average accuracy values are presented in Table 2.



	unigram	bigram	trigram	1-2-gram	1-2-3-gram
DS1	41.27	47.91	44.77	69.12	78.6
DS2	52.43	42.52	37.05	58.67	63.64
DS3	51.53	40.87	34.38	55.56	58.78
DS4	50.89	39.41	31.71	53.78	53.72
DS5	49.04	37.82	29.04	51.22	52.18
DS6	48.66	36.26	26.37	44.49	51.44
DS7	47.67	34.71	23.70	40.25	51.01

Fig. 5. Logistic Regression classification average accuracy without POS.



	unigram	bigram	trigram	1-2-gram	1-2-3-gram
DS1	41.47	56.15	56	68.19	78.29
DS2	40.45	50.41	47.38	59.39	64.25
DS3	40.14	49.2	46.01	55.85	59.52
DS4	40.5	47.48	44.13	52.1	54.27
DS5	39.88	46.75	42.64	51.41	53.78
DS6	39.84	46.27	42.23	50.73	52.03
DS7	40.06	46.35	42.06	50.42	51.69

Fig. 6. Logistic Regression classification average accuracy with POS.

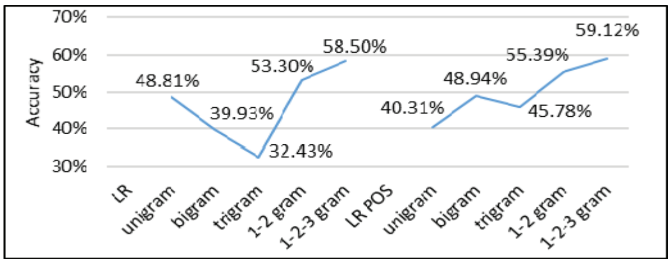


Fig. 7. Average Accuracy.

Findings indicate that LR classifier with POS has higher classification accuracy (~13%) if using bigram and trigram in comparison to LR classifier without POS and applying bigram and trigram model. Finding also indicate that using unigram with POS has smaller average accuracy in comparison to the model without POS.

TABLE II. SUMMARY RESULTS

n-gram	Without POS	More or less	With POS	Difference
unigram	48.81	>	40.31	-8.51
bigram	39.93	<	48.94	9.02
trigram	32.43	<	45.78	13.35
1-2 gram	53.30	<	55.39	2.09
1-2-3 gram	58.50	<	59.12	0.62
			Average	**3.31**

IV. CONCLUSIONS

A comparison of the LR classification method with and without POS is presented in this paper.

Experimental results have shown that applied LR multi-class classification method for product-review data using POS features has in average (3.31%) higher classification accuracy.

Experimental results have shown that average classification accuracy of LR classifier without and with POS tagging decreases with bigram and trigram models (without POS: min 23.70%, max 47.27%; with POS: min 42.06%, max 56.15%) and increases, if a combination of uni/bi/tri-gram models are applied (without POS: min 40.25%, max 78.6%; with POS: min 50.42%, max 78.29%), in comparison to the results of unigram (without POS: min 41.27%, max 47.67%; with POS: min 41.47%, max 40.06%).

Following the comparative analysis, it can be indicated that overall classification accuracy with combination of uni/bi/tri-gram models increase the average of classification accuracy, but it is decreasing when increasing the size of the data set.

REFERENCES

[1] E. Alpaydin, Introduction to Machine Learning, Second Edition ed., Cambridge: The MIT Press, 2010, pp. 1 - 3.

[2] T. Joachims, "Text categorization with support vector machines: learn-ing with many relevant features," in *Proceedings of ECML-98, 10th Euro-pean Conference on Machine Learning*, 1998.

[3] Dumais, J. Platt, D. Heckerman, M. Sahami, "Inductive learning al-gorithms and representations for text categorization," in *Proceedings of the seventh international conference on Information and knowledge manage-ment*, Bethesda, Maryland, USA, 1998.

[4] McEnery, A. and Wilson, A., Corpus Linguistics, 2nd ed., Edinburgh: Edinburgh University Press, 2001.

[5] Stefanie Dipper, "Theory-driven and corpus-driven computational linguistics, and the use of corpora," in *Corpus Linguistics: An International Handbook, Volume 1* , Berlin, Mouton de Gruyter, 2009, pp. pp. 68-96.

[6] Nitin Indurkhya, Fred J. Damerau, Handbook of Natural Language Processing, 2nd edition, 2010.

[7] Grefenstette, G. and Tapanainen, P., "What is a Word, what is a Sentence? Problems of Tokenization.," in *International Conference on Computational Lexicography*, Budapest, 1994.

[8] Choi, Freddy Y. Y., "Advances in domain independent linear text segmentation," in *Proceeding NAACL 2000 Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, Stroudsburg, 2000.

[9] C. Trim, "The Art of Tokenization," IBM, 2013. [Online]. Available: https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en. [Accessed 28 August 2016].

[10] V. Daudaravičius, Collocation segmentation for text chunking, Kaunas: Vytautas Magnus University, 2012.

[11] Yorick Wilks and Mark Stevenson, "The grammar of sense: Using part-of-speech tags as a first step in," *Journal of Natural Language Engineering,* vol. 4, no. 2, p. pp. 135–144, 1998.

[12] Farah Benamara, Carmine Cesarano, Antonio Picariello, Diego Reforgiato, and V. S. Subrahmanian, "Sentiment analysis: Adjectives and adverbs are better than adjectives alone," in *In Proceedings of the International Conference on Weblogs and Social Media* , 2007.

[13] Tetsuya Nasukawa and Jeonghee Yi, "Sentiment analysis: Capturing favorability using natural language processing," in *In Proceedings of the Conference on Knowledge Capture*, 2003.

[14] Janyce M. Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin, "Learning subjective language," in *Computational Linguistics*, 2004, p. pp. 277–308.

[15] Bo Pang and Lillian Lee, Opinion mining and sentiment analysis, vol. 2, Now Publishers Inc, 2008, pp. pp.21-22.

[16] M. Porter, "The Porter Stemming Algorithm," [Online]. Available: https://tartarus.org/martin/PorterStemmer/. [Accessed 23 Sep 2016].

[17] Sang, Erik F. Tjong Kim, "Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition," in *Proceedings of CoNLL-2002*, Taipei, Taiwan, 2002.

[18] Steven Bird, Ewan Klein, Edward Loper, Natural Language Processing with Python, O'Reilly Media, 2009.

[19] J. McAuley, C. Targett, J. Shi, A. van den Hengel, "Image-based recommendations on styles and substitutes," *SIGIR,* 2015.

[20] M. Rennie, L. Shih, J. Teevan, D. Karger, "Tackling the Poor Assumptions of Naive Bayes Text Classifiers," 2003.

[21] The Apache Software Foundation, "MLlib is Apache Spark's scalable machine learning library," [Online]. Available: http://spark.apache.org/mllib/. [Accessed 8 March 2016].

[22] M. Seddon, "Natural Language Processing with Apache Spark ML and Amazon Reviews," 2015. [Online]. Available: https://mike.seddon.ca/natural-language-processing-with-apache-spark-ml-and-amazon-reviews-part-1/. [Accessed 10 March 2016].

[23] Marina Sokolova, Guy Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management,* vol. 45, p. 427–437, 2009.

[24] M. Caraciolo, "Machine Learning with Python - Logistic Regression," ARTIFICIAL INTELLIGENCE IN MOTION, 2011. [Online]. Available: http://aimotion.blogspot.lt/2011/11/machine-learning-with-python-logistic.html. [Accessed 5 September 2016].

[25] Gerald. J. Popek, Robert P. Goldberg, "Formal Requirements for Virtualizable Third Generation Architectures," *Communications of the ACM,* vol. 17, no. 7, p. 412–421, 1974.