DOI: http://dx.doi.org/10.4314/star.v3i3.25
ISSN: 2226-7522(Print) and 2305-3372 (Online)
Science, Technology and Arts Research Journal
Sci. Technol. Arts Res. J., July-Sep 2014, 3(3): 147-155
Journal Homepage: http://www.starjournal.org/

Original Research

A Word Sense Disambiguation Model for Amharic Words using Semi-Supervised Learning Paradigm

Getahun Wassie[1*], Ramesh Babu P[1], Solomon Teferra[2] and Million Meshesha[2]

[1] College of Engineering and Technology, Wollega University, Post Box No: 395, Nekemte, Ethiopia

[2]School of Information Science, Addis Ababa University, Post Box No: 1176, Addis Ababa, Ethiopia

Abstract

The main objective of this research was to design a WSD (word sense disambiguation) prototype model for Amharic words using semi-supervised learning method to extract training sets which minimizes the amount of the required human intervention and it can produce considerable improvement in learning accuracy. Due to the unavailability of Amharic word net, only five words were selected. These words were *atena* (አጠና), *derese* (ደረሰ), *tenesa* (ተነሳ), *bela* (በላ) and *ale* (አለ). A separate data sets using five ambiguous words were prepared for the development of this Amharic WSD prototype. The final classification task was done on fully labelled training set using Adaboost, bagging, and AD tree classification algorithms on WEKA package.

Article Information

Article History:
Received : 25-06-2014
Revised : 19-09-2014
Accepted : 26-09-2014

Keywords:
Ambiguity
Bootstrapping
Word Sense disambiguation

***Corresponding Author:**
Getahun Wassie
E-mail: getahunit@yahoo.com

INTRODUCTION

Nowadays, the advancement of information technology has given birth to the internet that results huge collection of information to address the information need of the society. Although the advantage of the technology keeps up, natural language ambiguities become challenging problems due to scarcity of natural language processing systems in many languages. Amharic has been one of the under-resourced languages both in terms of electronic resources and natural language processing tools to access favourable conditions that information technology has brought (Atelach Alemu and Lars, 2010).

To address these challenges and create background solution for natural language processing systems for the language, word sense disambiguation (WSD) is one of the techniques proposed to reduce or avoid word ambiguities (Ravi Som Sinha, 2008). The application of WSD has great utility of fields including information retrieval, information extraction and machine translation (Ravi Som Sinha, 2008).

Natural language is highly ambiguous and computers do not have the common sense that enables real sense understanding (Maurice Van Keulen and Mena B. Habib, 2011). They do not also have the knowledge that humans have when they communicate, unless the language is represented with natural language representation mechanisms. To a human being, the intended meaning of the sentence with ambiguous word is clear depending on the circumstance but for a computer it is far from obvious.

When we are dealing with the complexity of the language, one of the difficulties to be considered is the word ambiguity (Maurice Van Keulen and Mena B. Habib, 2011). This is because many natural language possessing systems are built considering words; specifically content bearing words.

The meaning of a word may vary significantly according to the context in which it is used i.e. the meaning of a word depends on its context of occurrence (Georgios *et al*., 2010). WSD is defined as the task of resolving ambiguities in a text that results from the inherent polysemous nature of a language (Ravi Som Sinha, 2008). It is the task of assigning one of several possible sense labels to an ambiguity word (Samuel Brody, 2009). It is an association of a given word in a text or discourse with a sense which is distinguishable from other meanings potentially attributable to that word (Adam Kilgarriff, 2003). For instance, let us take a word from Amharic dictionary bela (በላ), it has two meanings. The English equivalent senses would be “to eat” or “to say”; therefore, this word has word sense ambiguity problem.

WSD was first formulated as a distinct computational task during the early days of machine translation in the late 1940s. It was known as one of the oldest problems in computational linguistics (Eneko and Philip, 2006). Today, Word sense disambiguation systems can be developed using two main approaches. Most current WSD approaches can be characterized as knowledge based or corpus-based. Knowledge based uses external sources as otology information. Corpus-based is machine learning approach that is farther divided into supervised, semi-supervised and unsupervised learning methods (Solomon, 2010).

Supervised learning method is a classification technique which requires a predefined human annotated data with class where as unsupervised learning paradigms which require unlabelled data to cluster based on data similarity.

Both supervised learning and unsupervised learning paradigms have their own limitations to achieve good classification and clustering results, but their limitations are complementary (Qiong *et al*., 2013). Supervised method requires external teachers to label the training data which is laborious (expensive), subjective and time taking. On the other hand, the unsupervised method yields significantly lower accuracy and produce results that are not satisfying for many applications and often derive sets of word senses that are not intuitive to humans (Bartosz and Maciej, 2013) due to the absence of human intervention for label annotation.

Semi-supervised learning method falls in the middle between unsupervised learning and supervised learning that make use of both a small amount of labelled and a large amount of unlabelled data for training(Bartosz and Maciej, 2013). It is an interesting method that considers how to avoid as many limitations as possible without losing their major advantages. One possible advantage is to achieve better classification result with less effort.

Amharic is the official language of the federal government of Ethiopia. It is mother tongue for more than 19 million and second language for over 5 million people. It is the most used language for information storage and media communication purposes in the country. There are many electronically published Amharic documents in the form of articles, news, researches, reports and WebPages etc.

Natural language dependent applications have been developed by understanding words, statements, phrases and etc. WSD has been applied on foreign languages to develop applications such as machine translation, information retrieval, text summarization, text classification, and question answering, speech recognition, information extraction and text mining (Ravi Som Sinha, 2010). WSD has the role of avoiding word ambiguities.

word ambiguity problem hinders to develop WSD based applications using this language i.e. highly polysemous words with Subtle sense distinctions still pose major challenges for automatic systems. Solving this problem of the language can initiate the development of WSD based applications for the language such as information retrieval systems.

Even though WSD has many applications and usage, almost no WSD systems have been done for Amharic, except a few WSD prototype models recently.

Solomon (Solomon, 2010) tested only supervised learning approach for Amharic WSD. However supervised machine learning approach of WSD performs better by human intervention, but it has limitations of knowledge-acquisition bottleneck i.e. it requires manually labelled sense examples which takes much time, very laborious and therefore very expensive to create when the corpus size increases. Solomon(Solomon Assemu, 2011) tested only unsupervised machine learning method that deals with grouping of contexts for the given word that express the same meaning without proving explicit sense labels for each group. Even though it avoids human intervention, is inexpensive and extremely scalable, it has worse performance than that of supervised one because it relies on less knowledge of sense annotation.

Since the supervised and unsupervised machine leaning methods have their drawbacks, the semi-supervised method narrows the gap of those methods by making use of labelled and unlabelled training data. In other tokens, it minimizes knowledge-acquisition bottleneck of supervised learning, and it improves poor performance of unsupervised learning. One of its main differences from the previously tested Amharic WSD models is the existence of training data from some labelled and many unlabelled datasets.

Current English WSD systems based on semi-supervised method generate attractive results i.e. unlabelled data can bring significant improvement in WSD accuracy (Tanah *et al*., 2011). It is therefore the concern of this study to apply semi-supervised learning techniques for a designing Amharic WSD prototype model for Amharic texts.

MATERIALS AND METHOD

Method of Semi-supervised learning

Current word sense disambiguation (WSD) systems are based on supervised learning methods which is still limited in that it does not work well for all words in a language. One of the main reasons is the lack of sufficient labelled training data that require expertise. Even though one can always label more examples to achieve better performance on a particular dataset but the expense can be uncomforted (Yarowsky, 2008). Since this method needs much effort to label the unlabelled instances, unsupervised learning becomes an alternative technique. This method does not require human effort for corpus annotation. However, it provides less performance.

Semi-supervised learning has become a potential machine learning method and applied for real world learning tasks ranging from data mining to medical diagnosis today. Supervised and unsupervised machine learning has been combined to balance their extreme drawbacks (Solomon, 2010). The unsupervised part is usually applied first to the data in order to make some assumptions about the distribution of the data, and then these assumptions are reinforced using a supervised one (Glenn Fung, 2001). Empirical results show that unlabelled data can bring significant improvement in WSD accuracy using semi-supervising learning (Thanh *et al*., 2012).

Semi-supervised learning is initially motivated by its practical value in learning better because small seed examples are used to train an initial model using any supervised algorithms (Rohan Sharma, 2008). In many real world applications, it is relatively easy to acquire a large amount of unlabelled data (Hendrik *et al*., 2013). Considering the availability of this large amount of unlabelled data, together with some labelled data (learning from partially labelled data), to build better classifiers with less human effort gives higher accuracy. Therefore Semi-supervised learning methods have received great attention both in theory and in practice recently.

Generally, semi-supervised learning is to automatically label the unlabelled examples using a small number of manually labelled examples as seeds. In this research, we incorporate unlabelled data by combining some seeds instances directly into the data representation (features), so that unsupervised algorithms can be directly applied for clustering based on instances similarity. Supervised algorithms can also be applied for classification after the unlabelled data are given their labels using the cluster labels found during clustering.

Bootstrapping

Bootstrapping is a general purpose sample-based statistical method which consists of drawing randomly with replacement from instances of training set (Artur, 2007). It needs only a few seeds instead of a large number of training examples unlike pure supervised approaches. It builds continuous optimization of the trained model until it reaches convergence (Xiaojin and Hyoil, 2008). One of the first uses of unlabelled data was to bootstrap an existing supervised learner using unlabelled data iteratively, this is also called "Self-training" (Pavan Kumar, 2010). Self-training is a common technique for Semi-supervised learning. In self training a classifier is first trained with the small labelled data. The created classifier is then used to classify the unlabelled data; here the classifier uses its own predictions to teach itself (Xiaojin, 2008).

The unlabelled data is labelled using a supervised learner trained on the labelled data, and the training set is augmented by the most confident labelled samples. In other tokens, typically the most confident unlabelled points, together with their predicted labels, are added to the training set.

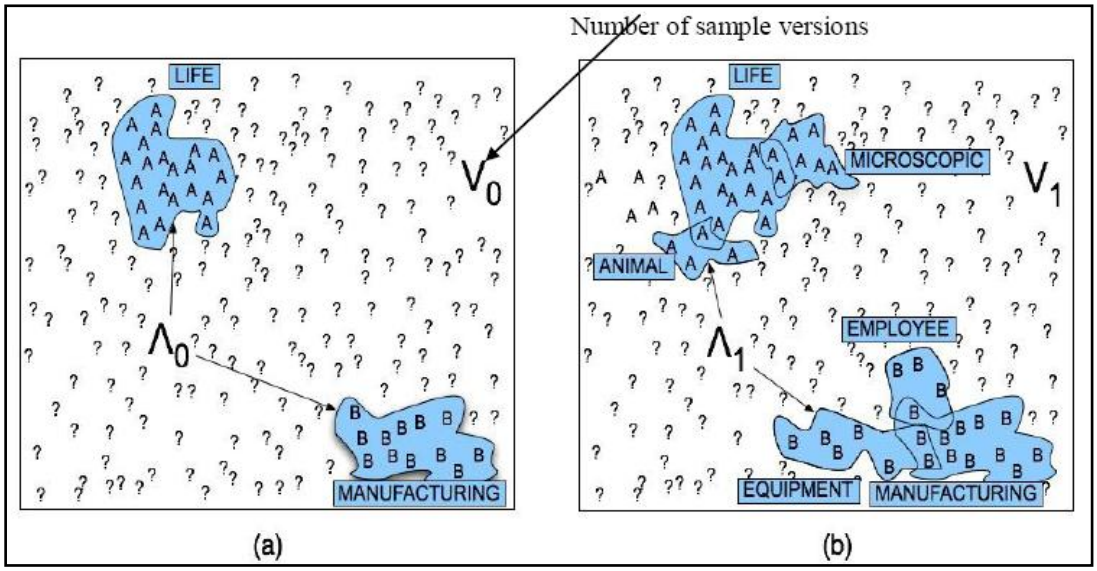

Figure 1: Bootstrapping Classification (Adapted from, Yarowsky, 2008)

Bootstrapping algorithms work by iteratively classifying unlabelled examples and adding confidently classified examples into labelled dataset using a model learned from augmented labelled dataset in previous iteration .

Bootstrap Algorithm

Input: Z = {z1, z2, . . . , zN}, with zi = (xi, yi) as training set.

B, number of sampled versions of the training set.

Output: S(Z), statistical estimate and its accuracy.

for n=1 to B

a) Draw, with replacement, L $\leq$ N samples from the training set Z, obtaining the nth sample Z^{*n}.

b) For each sample Z^{*n}, estimate a statistic $S(Z^{*n})$.

2. Produce the bootstrap estimate S(Z), using $S(Z^{*n})$ with n = {1, . . . ,B}.

3. Compute the accuracy of the estimate, using the variance or some other criterion

To depict the bootstrap algorithm with graphical idea the algorithm starts with the training set Z, obtaining several bootstrap samples (Z^{*n}). For each bootstrap samples, statistical measure can be computed S (Z^{*n}).

Some bootstrapping algorithms are discussed hereunder:

Bagging: A Bootstrap sample is produced by uniformly sampling m instances from the training set with replacement. Bootstrap samples B1, B2… BT is generated and a model C_i is built from each bootstrap sample B_i. A final model C* is built from C1; C2… CT whose output is with predicted class for unlabelled data. Bagging averages this prediction over a collection of bootstrap samples (Eric and Kohavi, 1998).

Input: training set S, Inducer $\mathcal{I}$, integer T (number of bootstrap samples).

1. for $i = 1$ to T {
2. $S' =$ bootstrap sample from S (i.i.d. sample with replacement).
3. $C_i = \mathcal{I}(S')$
4. }
5. $C^*(x) = \arg\max_{y \in Y} \sum_{i:C_i(x)=y} 1$ (the most often predicted label y)

Output: classifier C^*.

Figure 2: The Bagging Algorithm (adopted from Eric and Kohavi, 1998)

Adaboost: The Adaboost algorithm generates a set of classifiers and votes them until the final classifier is achieved like bagging does. The drawn sample in Adaboost case is based on weighted samples rather than choosing randomly. The simplest way to generalize to semi-supervised problems consists on defining a loss/margin for unlabelled data. This generalization to unlabelled data should not affect the loss for labelled examples. During visualization, the missing labels are due to the absence of class information. The pattern belongs to a class, but the class is unknown. Difficult instances get more attention for classification.

This algorithm is able to work efficiently in very high dimensional feature spaces, and has been applied, with performance success, to a number of practical applications (Gerard Escudero Bakx, 2006).

ADtree: Since ADtree has a reduced training cost and a very much reduced computation cost with respect to Adaboost, it is comparably applied for real-time applications as Adaboost. Adatree becomes more similar with Adaboost when the input data has little noise i.e. it is strong to avoid the data over-fitting that is why it is applied directly in real-time applications (Etienne, 2002).

Word Ambiguity in Amharic Language

Amharic is one of the languages that have their own unique writing system. The language has 33 consonants and 7 vowels. Each of the 33 consonants has seven orders in horizontal position that provide 151 Fidel patterns. Among the orders, six of them are consonant-vowel combinations while the 1th is consonant itself (Dawit, 2003). The first symbol of the orders is the basic symbol; the other orders are derived from first order symbols by coupling different vowels. The following script order in table 1 describes about Amharic Fidel "ሀ" and "ለ" in Power Geez Unicode1 font style to clarify the idea with comparable English script.

Table 1: An example of Amharic Fidel

vowel	-	u	i	a	y	E	o
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ

The table denotes the consonant with inherent vowels which are consistently modified to indicate vowels or, in some cases, the lack of a vowel (Saba, 2007). Besides these fidels, Amharic has 16 labialized consonants such as ሏ (*HWa*), ሟ (*mWa*), ሯ (*rWa*) etc. The total Fidel in Amharic becomes 135 distinct symbols today. These Fidel is written left-to-right unlike Arabic (Saba, 2007). Although Amharic scripts have been used by the speakers, the writing system has problem of standardizations. One of the problems is that the presence of different fidels having similar pronunciation or function. For instance the Amharic word "*srat*" can also be written as ስርአት, ሥርአት, ስርዓት and ሥርዓት representing the same word with different characters. Writers may use those Fidel in words interchangeably but in Amharic, the variant of these Fidel change meaning of a word that bring word ambiguity.

Ambiguities in Amharic language arise mainly due to symbol redundancy such as "አ", and "ዓ", are similar letters used interchangeably, and visual similarity or different character, such as "ፕ" and "ፐ", they can be also used interchangeably as various Amharic words, thus, forming different forms of spelling for the same word. The different forms of spelling make computers to consider a word as different words (Alemayehu, 2010).

Amharic Punctuation Marks and Capitalization

The Amharic writing system holds 17 punctuation marks in Addition to alphabets of which only a few of them are commonly used and have representations in Amharic software (Saba, 2007). These different Amharic punctuation marks are used for different functions. The following are some punctuation marks commonly used today (Tewodros *et al*., 2003).

Hulet Neteb(:), this mark used for separating words . For the modern Amharic, it is left except for hand writing purpose. Its place is almost completely taken over by white space. In English, words are separated by white space. This shows that modern Amharic borrows some writing style from English.

Arat neteb (::) is Sentence separator which is basic punctuation marks in Amharic writing system. It shows the end of a sentence, as a single dot implies end of a sentence in English.

Netela sereze(፡) which separates lists in Amharic text. It provides the equivalent function of comma in English.

Derib sereze (፤), which has the equivalent function of semi-colon of English. In addition to indigenous punctuation marks, some marks have been borrowed from foreign languages. For instance, the exclamation mark '*!*' and the question mark '*?*' are borrowed from English and used in Amharic language (Saba, 2007). There are also no upper and lower cases like English.

The removal of punctuation marks increases the effectiveness and efficiency of natural language processing systems (Atelach Alemu and Lars Asker, 2010) as stemming and stop word removal does. Doing this makes WSD prototype model efficient and more accurate. Like punctuation removal, stemming and stop word removal increases WSD systems` performance. Atalech et.al (Lars Asker *et al*., 2010) defines stemming as "a technique whereby morphological variants are reduced to a single stem". For languages like Amharic with very rich morphology is intuitively assumed that stemming will have a positive effect for classification and related tasks.

Syntactic Structure of Amharic and its Morphology

Amharic has a complex morphology. Sentences in Amharic are often short in terms of the number of words they are formed (Solomon *et al*., 2007). This nature of the language makes the window size (bag of context words) narrow. In other token, context words surrounding the word have more advantage for disambiguation purpose in WSD area.

Studying morphological aspects of languages helps to distinguish between lexical components of words which are accountable for the semantics of the words and grammatical words. Verbs are morphologically the most complex word class in Amharic with many inflectional forms. A substantial set of words in other word classes are derived primarily from verbs. Prepositions, articles, pronouns and conjunctions are often or always bound to other classes of words (38, 43). Even though Amharic sentences are short, it is morphologically rich with inflectional and derivational variants. Some lexeme may have identical forms and may be mapped to different forms (Saba, 2007). Consider this Amharic verb, *degeme*"ደገመ". Each of the following verbs has the prefix and suffix that indicate different morphologies based on the verb root.

yIdegmal 'ይደግማል'
yId_eg_emal 'ይደገማል'
yasdeg_Imal 'ያስደግማል
yIdegag_Imal 'ይደጋግማል'.

Amharic morphology creates word ambiguity in that two different words have same suffix or prefixes, they may have the same forms. For example, let us take two different Amharic words *bere*(በሬ) and *ber*(በር), when these words become plural, addition of suffix *och*(ኦች) changes the two words into one word as beroch (በሮች). Now, this word is ambiguous since it can be interpreted for two meanings. Syntactically, Amharic is an SOV language i.e. subject + object+ verb (Daniel, 2003). For example the sentence in English, " the boy ate injera" can be written in Amharic as "the boy injera ate"(ልጁ እንጀራ በላ::).we know that the Amharic word *bela*(በላ) has two meanings. One is "eat", and the other is "say". So it is an ambiguous word. Disambiguation can be performed to identify what the sentences are talking about after considering neighboring words (ልጁ and እንጀራ). This example shows the nature syntactical forms of Amharic. Ambiguities raised due to syntactical and other types are discussed on sections.

Amharic WSD Corpus Preparation

WSD systems need well organized datasets for training to make their accuracy attractive. The proposed WSD prototype used corpus to extract a lot of relevant words from it for disambiguation purpose. To prepare the corpus from its original text and avoid expert for translation, we prepare WSD corpus containing 5 words assuming that it is enough to develop the proposed prototype model. The words are selected from an Amharic dictionary (Amharic Dictionary). Table 2 shows the selected ambiguity words with their corresponding number of sentences.

Ambiguous words	Senses	#sense examples	Total
Atena(አጠና)	Strengthen	111	215
	Study	104	
Derese(ደረሰ)	Reach	100	207
	Mature	107	
Tenesa(ተነሳ)	Stand	100	200
	Cause	100	
Ale(አለ)	Say	107	208
	Live/present	101	
Bela(በላ)	Eat	100	201
	Speak	101	
		Total	1031

Table 2: Ambiguity Words with their Senses

A total of 481 sentences were acquired for each senses of ambiguous word. The set of surrounding words to the left and right of the target word in the context, called window, is used for disambiguation. The target word is identified and written at the middle of sentences. We prepared window size of 10 considering that many WSD systems in many languages including English use Window size of 10-10 for disambiguation task. Let us explain it using the following example which is extracted from the corpus.

… hzib kadar eskedar landnetu blo **tenesa** Esayas Afworke selam felaga…

10 left context words 10 right context words

In the above example, the target word "*tanasA*" which is the ambiguous word and its word class provides sense "stand" in this context. 10 right and 10 left surrounding contexts of the target word are shown in the example. The rest six contexts should be included from the right and the left.

If a sentence that contains target word holds 20 contexts (10+10 from above window) including the target word, it can be used as instance of data unless context words from the previous and next sentences (adjacent sentences in the corpus) are taken to fulfil the window. Doing this helps to protect missing values that negatively impact the performance of the prototype.

Proposed System Architecture

The architecture of the proposed prototype system is as shown in figure 3. The system takes sentences or texts containing the ambiguous words as an input. The texts were pre-processed to make the system efficient and effective in its performance. Then the supervised and unsupervised learning algorithms (semi-supervised) are applied to build WSD model from a mixture of labelled and unlabelled training set. The next task is evaluation of the built model and performance status of the model. The detailed pre-processing and techniques of the processes are presented in the next subsections.

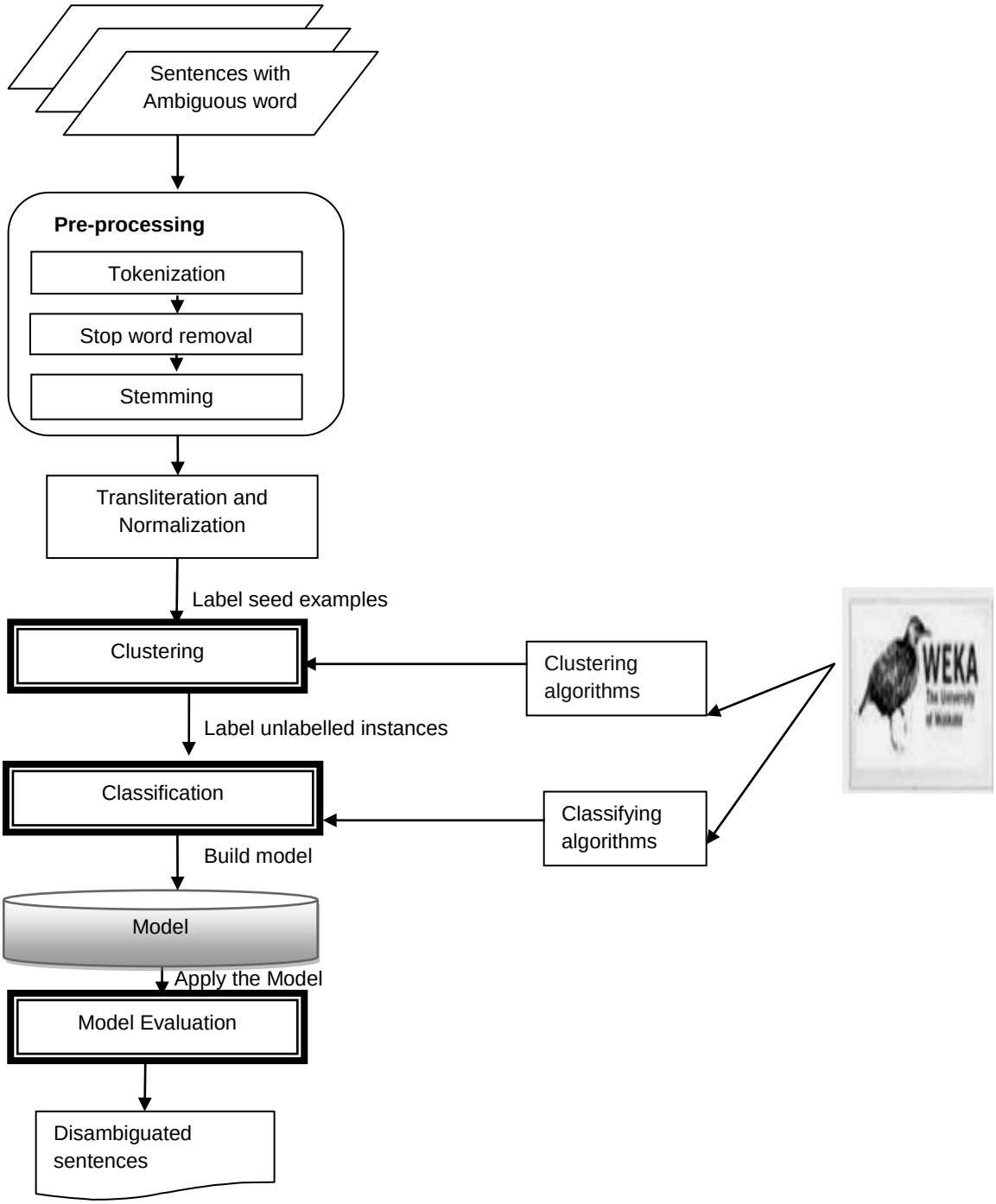

Figure 3: Architecture of Amharic WSD prototype

Preprocessing

As most NLP systems, a preliminary preprocessing of the input text is needed. Texts (sentences) preprocessing is a primary step to load the instances of dataset into machine learning tool (WEKA) to develop WSD model for the study. The preprocessing task comprises tokenization, stop word removal, stemming and normalization. The prepossessing part of the WSD prototype was accomplished using python 1.1.3 software. We have used a python program (code) which we make it in line with our preprocessing tasks.

Normalization: It is the process of converting words having different form of writing into a normalized form. Normalization techniques can be a case folding that involves the conversion of between sentence cases, capital case, and title case, uppercase in English (Suneetha and Sameen, 2011). But Amharic language does not need such case conversion since the language is normalized by its nature i.e. no capitalization in Amharic.

Since one Fidel has many different symbols in Amharic, normalizing similar symbols with one symbol is an important task to avoid ambiguity. But, the good thing for our dataset normalization is that the selected SERA system (ethiop) embedded the normalization capability during transliteration (Yacob, 1996).

Transliteration: After the pre-processing tasks have been done on collected texts; transliteration task was accomplished from Amharic to Latin characters. Doing transliteration makes compatible with the machine learning tools (WEKA) selected for the experiments.

Transliteration is the representation of characters of one language by corresponding characters of another language. The characters of Amharic texts are represented using a variety of character forms. SERA (System for Ethiopic Representation in ASCII) are a convention for the transcription of fidel (Ethiopic script) into the seven bit ASCII format.

In this research, we transliterated all Amharic texts into Latin using SERA representation using g2 commands on Linux platform. Amharic fidels are transliterated into Latin alphabets based on ethiop fidel forms of SERA system (Yacob, 1996). For example the Amharic fidel '*ሀ*'can be transliterated into its equivalent Latin characters 'hA' in ethiop fidel form.

SERA, is case sensitive, i.e., upper and lower cases of the English alphabet representing different symbols in the Amharic alphabet. Therefore, other Amharic alphabets that have the same meaning and sound with different form are transliterated to the same form without affecting the meanings of words (Solomon, 2010). In other words, normalization of characters was done using SERA system. After the data have been pre-processed and transliterated, clustering and classifying activities of data mining techniques were performed using standard algorithms.

Techniques

This section presents the technique applied for this study. Given a set *U* of unlabelled documents and some L of labelled seed examples, our technique has four steps.

Step 1: Selecting Representatives seed examples for each class

Step 2: Cluster the documents in U and L using "class to cluster" mode

Step 3: Perform feature selection on fully labelled dataset and feature extraction.

Step 4: Build the final classifier.

RESULTS

Semi-supervised learning which combines supervised and unsupervised methods by exploiting unlabelled examples to improve learning performance (Xiaojin and Hyoil, 2008). It uses both labelled and unlabelled training data together to improve WSD accuracy (Bartosz and Maciej, 2013). The combination of many unlabelled data and some labelled seed examples improves performance with less effort.

The experiment of semi-supervised methods using selected algorithms was conducted on the five Amharic WSD datasets following semi-supervised clustering assumption. The experimental result is shown in table 3.

Table 3: Results Using Semi-supervised learning bootstrapping (3-3 window)

Training words set	Adaboost	bagging	ADtree
Atena	93.95%	88.37%	97.21%
Derese	90.34%	81.16%	95.65%
Tenesa	82%	81.50%	81.50%
Ale	68.27%	68.27%	76.92%
Bela	89.95%	86.93%	90.95%
Average	**84.90%**	**81.25%**	**88.45%**

Semi-supervised learning method using bootstrapping improves purely supervised method such as naïve bayes. Applying Bootstrapping algorithms on WSD task performance gives good results in accuracy (Yarowsky, 2008). Likewise, in our research we got the average performance results of Adaboost, Bagging and ADtree algorithms are 84.90%, 81.25% and 88.45%.

The visualization of performance results on datasets using algorithms is shown figure 1.4 for more clarification.

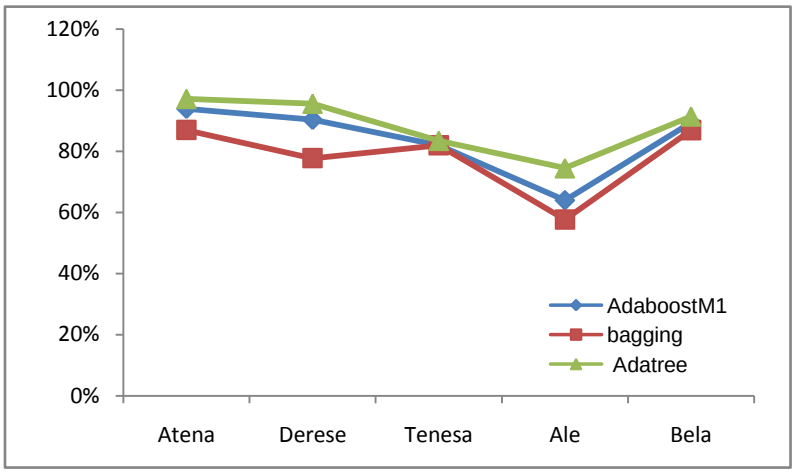

Figure 4: Performance Results of Semi-supervised Learning

DISCUSSION

Based on different classifying algorithms used, the optimal window size of Amharic word ambiguities is rotating around window size 2 or 3. Window size of 3 is also recommended for Amharic word disambiguation using supervised learning method.

Considering the variety of employed algorithms, the optimal window size determination was based on the performance of bootstrapping. We took the five optimal window sizes which are obtained from each word datasets using three algorithms. Three bootstrapping algorithms Adaboost, ADtree, and Bagging were tested for this purpose.

Adaboost is a variant of adaboost algorithm. Adaptive boosting is the weight of the data instead Of random sampling. It is a well-known method to build classifiers with attractive performance. We employed this algorithm on Amharic WSD datasets and achieved performance accuracy 83.90% in average as displayed in table 3.

ADtree is boosting Version of decision tree because early experimentation shows a reduced computation cost with respect to Adaboost. Adaboost yields better efficiency and tree provides good visualization of algorithms and tree structure. These good properties of Adaboost and decision tree are preserved in ADtree. Implementing this algorithm on Amharic WSD systems became effective that its average performance is 88.45%.

Bagging takes only a small training set from 5% to 20% of the original training data formed by random sampling with replacement. The size of these sub-sampled datasets is equal to the size of the original training set. We apply this algorithm on Amharic WSD prototype and we found that average performance of 81.25% accuracy along datasets.

Therefore, Amharic word ambiguities need 3 lemmas before the ambiguous word and 3 lemmas after using bootstrapping algorithms.

CONCLUSION

The concern of this research is word sense disambiguation which assigns one of several possible sense labels to an ambiguity word .WSD has been applied in many languages to be utilized in many applications areas such as machine translation, information retrieval, and information extraction to minimize word ambiguities.Although Amharic has many ambiguous words, we selected only five ambiguous words to build Amharic WSD prototype from Amharic dictionary. These words are ale, atena, bela, derese, and tenesa.

We conclude that Semi-supervised learning using bootstrapping algorithm performs better in our study. It is more adaptive on WSD for the Amharic. Specifically, Adtree, Adaboost and bagging are potential algorithms to be applied for Amharic WSD systems using semi-supervised learning methods. Considering our less datasets, we found that standard window size of three words before and three words after (3-3) the ambiguous word is considered to be enough for Amharic WSD model taking algorithm difference under consideration. Therefore, a window size of 3-3 can be a standard window size for Amharic WSD systems development.

REFERENCES

Adam Kilgarriff. (2003). Word sense disambiguation and parallel corpora, *University of Brighton and Lexicography Master Class*.

Alemayehu. (2010). Application of query expansion for Amharic information retrieval system, *M.Sc Thesis*, Addis Ababa University Ethiopia.

Artur Ferreira. (2007). Survey on Boosting Algorithms for Supervised and Semi-supervised Learning, *Institute of telecommunications*.

Atelach Alemu Argaw and Lars Asker. (2010). An Amharic Stemmer: Reducing Words to their Citation Forms, *M.S Thesis, Department of Computer and Systems Sciences*, Stockholm University, Sweden.

Bartosz Broda and Maciej Piasecki. (2013). Semi-supervised Word Sense Disambiguation Based on Weakly Controlled Sense Induction, *Proceedings of the International Multi-conference pp. 17–16,* Institute of Informatics Wroclaw University of Technology.

Daniel Gochel Agonafer. (2003). An Integrated Approach to Automatic Complex Sentence Parsing For Amharic Text, *M.S Thesis*, Department Of Information Science, Addis Ababa University.

Dawit Bekele. (2003). The Development and Dissemination of Ethiopic Standards and Software Localization for Ethiopia, *The ICT Capacity Building Programme of the*

Capacity Building Ministry of the FDRE and United Nations Economic Commission for Africa, Addis Ababa.

Eneko Agirre and Philip Edmonds. (2006). Word Sense Disambiguation: Algorithms and Applications*, University of the Basque Country*, Sharp Laboratories of Europe Limited.

Eric Bauer, Ron Kohavi. (1998). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants, *Machine Learning Research Centre*, Netherlands.

Ethiopian Language Research Institute-*Amharic Dictionary Center*, Addis Ababa, Ethiopia.

Etienne Grossmann. (2002). ADtree : boosting a weak classifier into a decision tree, *University of Kentucky*, Quality Street, Suite 412, Lexington, KY, 40507, USA.

Georgios Paliouras., Vangelis Karkaletsis. and Constantine D. Spyropoulos. (2010). Learning Rules for Large Vocabulary Word Sense Disambiguation, *Institute of Informatics and Telecommunications*, NCSR "Demokritos", Aghia Paraskevi Attikis Athens, Greece.

Gerard Escudero Bakx. (2006). Machine Learning Techniques For Word Sense Disambiguation, *PH.D Dissertation, universitat politμecnica de catalunya*, Barcelona.

Glenn Fung. (2001). A Comprehensive Overview of Basic Clustering Algorithms.

Hendrik K¨uck., Peter Carbonetto. and Nando de Freitas. (2013). A Constrained Semi-Supervised Learning Approach to Data Association, Dept. of Computer Science, University of British Columbia, Vancouver, Canada.

Lars Asker., Atelach Alemu Argaw., Björn Gambäck and Magnus Sahlgren. (2010). Applying Machine Learning to Amharic Text Classification, *Stockholm University and Swedish Institute of Computer Science*.

Maurice van Keulen and Mena B. Habib. (2011). Handling uncertainty in information extraction, *Ph.D Thesis, University of Twente,* Enschede, The Netherlands.

Pavan Kumar Mallapragada. (2010). Some Contributions to Semi-Supervised Learning, *M.S.Thesis*, Michigan State University.

Qiong Liu., Stephen Levinson., Ying Wu. and Thomas Huang. (2013). Interactive and Incremental Learning via a Mixture of Supervised and Unsupervised Learning Strategies**,** *Beckman Institute for Advanced Science and Technology, University of Illinois* at Urbana-Champaign.

Ravi Som Sinha. (2008). Graph-Based Centrality Algorithms For unsupervised Word Sense Disambiguation, *M.S. Thesis*, University Of North Texas.

Rohan Sharma. (2008). Word Sense Disambiguation for Hindi Language, M.Sc. Thesis, Thapar University, Patiala.

Saba Amsalu Teserra. (2007). Bilingual Word and Chunk Alignment: A Hybrid System for Amharic and English, *M.S Thesis*, University of Bielefeld.

Samuel Brody. (2009). Closing the Gap in WSD: Supervised Results with Unsupervised Methods., *Dissertation, Institute for Communicating and Collaborative Systems*, School of Informatics, University of Edinburgh.

Solomon Assemu. (2011). Unsupervised Machine Learning Approach for Word Sense Disambiguation to Amharic Words, *M.Sc. Thesis, School of Information Science*, Addis Ababa University.

Solomon Mekonen. (2010). Word Sense Disambiguation for Amharic Text: A Machine Learning Approach*,* M.Sc. Thesis, School of Information Science, Addis Ababa University.

Solomon Teferra Abate and Wolfgang Menzel. (2007). Syllable-Based Speech Recognition for Amharic, *University of Hamburg, Department of Informatics*. Vogt-Kölln-Strasse. 19, D-22517 Hamburg, Germany.

Suneetha, M., Sameen Fatima, S. (2011). Corpus based Automatic Text Summarization System with HMM Tagger. *International Journal of Soft Computing and Engineering* 1(3):118-123.

Tewodros Hailemeskel Gebermariam. (2003). Amharic Text Retrieval: An Experiment Using Latent Semantic Indexing (LSI) With Singular Value Decomposition (SVD), *M.S Thesis*, Department of Information Science ,Addis Ababa University.

Thanh Phong., Pham, Hwee., Tou Ng and Wee Sun Lee. (2012). Word Sense Disambiguation with Semi-Supervised Learning*, Department of Computer Science*, National University of Singapore.

Thanh Phong., Phamand Hwee., Tou Ng. and Wee Sun Lee. (2011). Word Sense Disambiguation with Semi-Supervised Learning, *Department of Computer Science*, National University of Singapore.

Xiaohua Zhou and Hyoil Han. (2008). Survey of Word Sense Disambiguation Approaches*,* Drexel University, 3401 Chestnut Street, Philadelphia, PA 19104.

Xiaojin Zhu. (2008). Semi-Supervised Learning Literature Survey, *Computer Sciences TR 1519*, University of Wisconsin – Madison.

Yacob, D (1996). System for Ethiopic Representation in ASCII (SERA*)*. [cited 2012 Accessed on April 12]; *Available from: http://www.abyssiniacybergateway.net /fidel/*

Yarowsky D. (2008). Unsupervised word sense, disambiguation rivaling supervised methods, Cambridge, MA: ACL-47, 1947, pp. 144-196.