Advances in Computational Linguistics

Research in Computing Science

Series Editorial Board

Research in Computing Science es una publicación trimestral, de circulación internacional, editada por el Centro de Investigación en Computación del IPN, para dar a conocer los avances de investigación científica y desarrollo tecnológico de la comunidad científica internacional. **Volumen 84,** noviembre de 2014. Tiraje: 500 ejemplares. *Certificado de Reserva de Derechos al Uso Exclusivo del Título* No. : 04-2005-121611550100-102, expedido por el Instituto Nacional de Derecho de Autor. *Certificado de Licitud de Título* No. 12897, *Certificado de licitud de Contenido* No. 10470, expedidos por la Comisión Calificadora de Publicaciones y Revistas Ilustradas. El contenido de los artículos es responsabilidad exclusiva de sus respectivos autores. Queda prohibida la reproducción total o parcial, por cualquier medio, sin el permiso expreso del editor, excepto para uso personal o de estudio haciendo cita explícita en la primera página de cada documento. Impreso en la Ciudad de México, en los Talleres Gráficos del IPN – Dirección de Publicaciones, Tres Guerras 27, Centro Histórico, México, D.F. Distribuida por el Centro de Investigación en Computación, Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othón de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738, México, D.F. Tel. 57 29 60 00, ext. 56571.

Editor responsable: *Grigori Sidorov*, *RFC SIGR651028L69*

Research in Computing Science is published by the Center for Computing Research of IPN. **Volume 84**, November 2014. Printing 500. The authors are responsible for the contents of their articles. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without prior permission of Centre for Computing Research. Printed in Mexico City, in the IPN Graphic Workshop – Publication Office.

Volume 84

Advances in Computational Linguistics

Alexander Gelbukh (ed.)

Instituto Politécnico Nacional, Centro de Investigación en Computación
México 2014

ISSN: 1870-4069

Instituto Politécnico Nacional (IPN)
Centro de Investigación en Computación (CIC)
Av. Juan de Dios Bátiz s/n esq. M. Othón de Mendizábal
Unidad Profesional "Adolfo López Mateos", Zacatenco
07738, México D.F., México

http://www.rcs.cic.ipn.mx
http://www.ipn.mx
http://www.cic.ipn.mx

Indexed in LATINDEX and Periodica / Indexada en LATINDEX y Periódica

Printing: 500 / Tiraje: 500

Printed in Mexico / Impreso en México

Editorial

This volume of the journal "Research in Computing Science" contains selected papers on the modern interdisciplinary research area related to the fields of humanities and computing science: computational linguistics (another term with more focusing on algorithms is natural language processing).

The papers were carefully chosen by the editorial board on the basis of the at least two reviews by the members of the reviewing committee or additional reviewers. The reviewers took into account the originality, scientific contribution to the field, soundness and technical quality of the papers. It is worth noting that various papers for this special issue were rejected.

This volume contains papers on various topics of computational linguistics and natural language processing, like ontology design, text clustering, machine translation, automatic morphological analysis, sentiment analysis and affective lexicon, advertising in social networks, and error analysis in pronunciation training.

I would like to thank Mexican Society for Artificial Intelligence (Sociedad Mexicana de Inteligencia Artificial), MICAI 2014 conference, Instituto Tecnológico de Tuxtla Gutierrez (Chiapas, Mexico), and Universidad Autónoma de Chiapas for their support during preparation of this volume.

The papers were collected and the reviewing process was organized using the system EasyChair.

Alexander Gelbukh
November 2014

Table of Contents

Page

Automatically Clustering Ontological Annotated Sentences to Detect Semantic Frames

Alexandra Moreira, Alcione Oliveira de Paiva, and Giorgio Torres

Departamento de Informática, Universidade Federal de Viçosa (UFV),
CEP 36570-000, Viçosa MG,
Brazil

xandramoreira@yahoo.com.br,
{alcione,torres.giorgio}@gmail.com
http://www.dpi.ufv.br

Abstract. Lexical databases of semantic frames have been shown to be useful in problems related to natural language processing. However, creation of such databases is a task that is time consuming and involves many manual steps. One of these steps is selection and grouping of sentences to identify frames. However, we advocate that if sentences were previously annotated with ontological information, this grouping could be executed automatically. In this article we present tests performed with clustering sentences containing the lexeme Travel (noun and verb). Tests showed that the use of clustering algorithms on ontologically annotated sentences is a promising step towards automating construction of semantic frames databases.

Keywords: Clustering sentences, ontological annotation, frame semantics, FrameNet.

1 Introduction

The frame semantics proposed by Charles Fillmore [6] is a theory which states that the meaning of a lexeme can only be known from the knowledge of the scene where it occurs. Based on this theory, lexical databases, called FrameNet, describing the predicate-argument structure elements in a given scene were developed [20]. Lexical databases of semantic frames have been shown to be useful in problems related to natural language processing [4] [8] [13]. However, creation of such databases is a task that is time consuming and involves many manual steps [20]. One of these steps is the selection and grouping of sentences to identify frames. According to [20], The core of the process is to search for corpus attestations of a group of words that the FrameNet developers believe to have some semantic overlap. After that step they divide these attestations into groups and afterwards, combine the small groups into large enough groupings to make reasonable frames at which point we may (equivalently) call the words targets, lexical units, or frame-evoking elements. As one can see, the process

is essentially manual, even with some auxiliary computational tools. However, automating this task is not a trivial process, since it requires a lot of common sense knowledge. We propose here to move up a step on the path to automate this process. We advocate that if sentences were previously annotated with ontological information, this grouping could be executed automatically. In this article we present tests performed with clustering sentences containing the lexeme Travel (noun and verb).

This article have the following structure: the next section presents the related work; section three succinctly presents the FrameNet; our proposal is presented in section four; section five presents the results and finally, section six presents the conclusions.

2 Related Works

Using semantic information to group or extracting information has been a subject widely investigated, nevertheless, no work that exploits corpus annotated with ontological types to perform groupings of sentences have been found. In [5] was presented a cooperative Machine Learning system which is able to acquire subcategorization verb frames with restrictions of selection and ontologies for specific domains from syntactically parsed technical texts in natural language. Texts and parsing may be noisy. The difference of this work is that the former extracts ontology instead of using it to detect the frame. Chow et al. [3] carried out a mapping between word-meanings (WordNet), frame-semantics (FrameNet) and world concepts captured by SUMO Ontology. The mapping provided a knowledge base for Semantic Role Labeling(SRL), identifying the appropriate range of possible semantic roles with respect to the event evoked by verb. In [1] was presented a research in Word Sense Disambiguation problem based on grouping noun representations of the senses. The proposal was based on the clustering of noun sense representations. In [10] is proposed an approach which utilizes ontology knowledge to automatically denote the implicit semantics of textual requirements. The authors state that "requirements documents include the syntax of natural language but not the semantics". They performed a semantic annotation of the requirements specification automatically and after this step is generated a domain model of the intended system. The common point with our work is the use of ontological annotation for analysis of sentences in natural language, however the scope and purpose differ widely from the present work.

3 The FrameNet

Frame Semantics arose as a response to the inability of traditional semantic to give account for different interpretations of lexical elements, such as explaining why it is not appropriate to characterize the Pope as a bachelor [9]. This is a classic example, used in several attestations [11] [16] [6] of the failure of the compositional semantic approach that defines a concept through minimum

and necessary conditions. In fact, to understand the concept evoked by the lexical unit bachelor, one need to understand a chain of interrelated conceptual structures, such as the institution of marriage in western world, the notion of the typical functions of a married man and when one person is able to exercise those functions. Only then is possible to properly apply the term "bachelor" to someone. This is true for the majority of lexemes in natural language. Lexemes whose meaning can only be understood by understanding the entire concepts involved (gestalt) and not by their individual analysis.

FrameNet is a lexical semantic database based on Semantic Frames and supported by evidence from corpora. The pioneer FrameNet was developed by the International Computer Science Institute in Berkeley under the leadership of Collin F. Baker, Charles J. Fillmore and John B. Lowe [2]. The project aims to record the semantic and syntactic combinatorial possibilities (valences) of each predicative word (names, adjectives and verbs) in each of its senses. The basic concepts underlying the FrameNet project are the concepts of *frames*, *relations* between frames, *lexical units* (LU) and *frame elements* (FE). A lexical unit (LU) is the pairing of a word with a meaning [20]. According to the same author, each sense of a polysemous word belongs to a different semantic frame. A LU evokes a frame. For example, the occurrence of the word *buy* in a sentence invokes the event of a commercial purchase captured by the **Commerce_buy** frame. Frame Elements (FE) are roles that occur in a given frame. For example, the frame **Commerce_buy** describes common situations involving roles such as buyer, goods, seller, location and money. By presenting a particular frame, the system displays a definition and a list of elements of frames, and for each FE is presented a set of annotated sentences, extracted from a *corpus*. Frames are interconnected, forming a system of frames. They are connected through semantic relationships, such as *inheritance*, *use*, *subframe* and *perspective*. This differentiates them from other lexical databases, such as the thesauri. Semantic relations are asymmetrical frames forming a directed graph.

As already mentioned lexical databases such as a FrameNet are useful in a variety of natural language processing applications. However, the construction of a FrameNet is essentially a manual work with the support of some computational tools. The proposal described below seeks to contribute to increase the degree of automation of the process.

4 The Proposal

Ontological information imposes contextual constraints and help establish the scene that is taking place. Sentences belonging to the same scene will contain the same ontological types or ontological types closely related. Adding of an annotation step to the FrameNet development process to add ontological type information is advocated by [15]. However the addition of such information is not a trivial task. There are some projects that address the task of ontological annotation, such as [17] and [21]. Here we report an use of this annotation layer with the objective of helping the grouping of sentences for extracting semantic

frames. Automatic annotation of ontological information is also being addressed within this project but there are still no published results. Fig. 1 summarizes the steps of the clustering process.

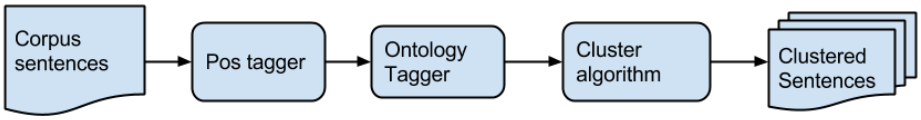

Fig. 1. The Clustering Process.

Part of speech (POS) annotation stage is important to help the ontology annotation step. After its lexical class had been identified is easier to identify the type of a term. To perform clustering the framework Weka was used. Weka [7] is a set of programs written in the Java programming language and is oriented carry out data mining and machine learning tasks. The Weka was developed by the University of Waikato in New Zealand and is an open source tool. The tool can be run directly or incorporated into other programs and provides tools for pre-processing, classification, regression, clustering and data visualization. The Framework has several clustering algorithms, which allows performing many tests within the same environment.

5 Results

To test the proposed system it was used a *subcorpus* of the *Corpus do Português* available for free access in the BRIGHAM YOUNG UNIVERSITY portal [1]. The subcorpus consists of sentences containing the lexeme "Travel", both the verb and the nominal in Brazilian Portuguese language. This subcorpus was used by [14] in the characterization of frame TRAVEL. In [9] the sentences were manually classified into *prototypical*, *quasi* and *metaphorical*. The prototypical class groups the typical sentences of the central meaning of the lexeme *travel*. That is: a displacement event to a particular locality executed by a conscious entity or group of entities, by themselves or by a transport means and for some purpose [2]. The *quasi* class groups the sentences that deviate in varying degrees from this central sense. The metaphorical class groups the sentences where the lexeme *travel* occurs in a metaphorical sense (e.g., time travel, spiritual, etc.). This is a good *corpus* to test whether the system will group in the same way sentences were manually grouped. 57 sentences were used as input to the system. 15 of these 57 sentences were previously classified as prototypical, 5 were classified as metaphorical, and 37 were previously classified as *quasi*.

[1] `http://corpus.byu.edu`
[2] `https://framenet.icsi.berkeley.edu/fndrupal/`

The ontology used was the SIMPLE-CLIPS ontology, (*Semantic Information for Multifunctional Plurilingual Lexica-Corpora e Lessici dell'Italiano Parlato e Scritto*) [12]. The SIMPLE-CLIPS ontology is based on *qualia* structure [18] and consists of semantic types organized through hierarchical and non-hierarchical conceptual relations. *Qualia* structure describes the nature of denotation through their fundamental attributes organized in formal, constitutive, telic and agentive dimensions. Ontological annotation was performed semi-automatically in [14].

Lexical items were annotated with the following semantic types: *human*, *vehicle*, *animal*, *abstract* and *local*. Occurrence or absence of these elements were used to create the vector space used by the clustering algorithm. Table 5 shows the attributes present in each sentence. The last attribute indicates the classification assigned by the human expert.

Table 1. Attributes present in each sentence.

vehicle local prototypical	vehicle local prototypical	vehicle abstract metaphorical
null quasi	vehicle local prototypical	human quasi
local prototypical	null quasi	human local prototypical
local prototypical	null quasi	animal quasi
human local prototypical	local prototypical	local quasi
vehicle prototypical	vehicle quasi	null quasi
human local prototypical	null quasi	human quasi
human vehicle prototypical	human quasi	human local prototypical
local prototypical	vehicle quasi	null quasi
null quasi	vehicle local prototypical	local quasi
null quasi	null quasi	vehicle quasi
animal abstract metaphorical	local quasi	animal quasi
local quasi	null quasi	null quasi
local quasi	animal quasi	vehicle local quasi
human quasi	abstract metaphorical	human quasi
human vehicle local prototypical	abstract quasi	null quasi
human quasi	local quasi	abstract metaphorical
human quasi	human quasi	null quasi
local quasi	abstract metaphorical	vehicle quasi

Those attributes lists were used as input for classification algorithms of Weka Framework. EM (expectation maximisation) algorithm was the one with best results. EM assigns a probability distribution to each instance which indicates the probability of it belonging to each of the clusters [7]. EM can decide how many clusters to create by cross validation, or one may specify *apriori* how many clusters to generate. It disagreed with the classification done by humans in 19%. This rate seems high at first glance, however, it is necessary to analyze this result more carefully. Fig. 2 presents part of the textual output of the Simple EM algorithm and Fig. 3 shows the plot of the Clustering.

6 Conclusion

Tests showed that the use of clustering algorithms on ontologically annotated sentences is a promising step towards automating the construction of semantic

```
=== Run information ===

Scheme:weka.clusterers.EM -I 100 -N -1 -M 1.0E-6 -S 100
Relation:     FrameTravel
Instances:    57
Attributes:   7
              human
              vehicle
              animal
              abstract
              local
Ignored:
              num
              frame
Test mode:Classes to clusters evaluation on training data

=== Model and evaluation on training set ===

Clustered Instances

0       6 ( 11%)
1      30 ( 53%)
2      21 ( 37%)

Log likelihood: -2.28633

Class attribute: frame
Classes to Clusters:

  0  1  2  <-- assigned to cluster
  0  2 13 | prototypical
  1 28  8 | quasi
  5  0  0 | metaphorical

Cluster 0 <-- metaphorical
Cluster 1 <-- quasi
Cluster 2 <-- prototypical

Incorrectly clustered instances :    11.0     19.2982 %
```

Fig. 2. Part of the textual output of the algorithm (Simple EM - expectation maximisation).

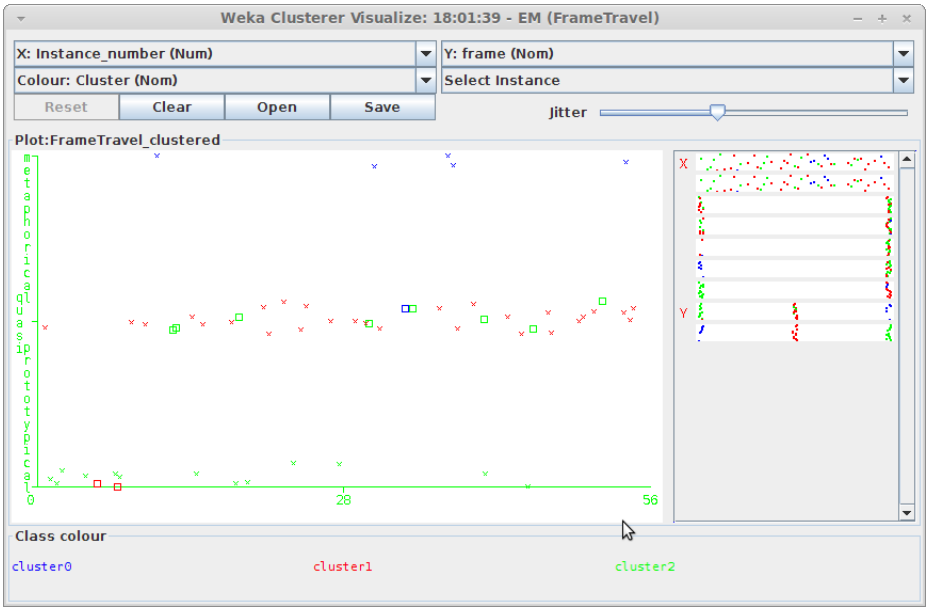

Fig. 3. Plot of the clustering (Simple EM - expectation maximisation).

frames databases. In typically sentences related to the frame it can be noted a reasonable degree of accuracy in two classical clustering algorithms. The disagreements are most common in sentences with few annotation or difficult to be framed even by people. The use of more accurate ontological types annotation algorithms should lead to better results. As future work, we are analyzing the semantic annotator Wmatrix [19] to enable a broader analysis of larger *corpus*.

Acknowledgments. This research is supported in part by the funding agencies FAPEMIG, CNPq and by the Gapso company.

References

1. Anaya-Sánchez, H., Pons-Porrata, A., Berlanga-Llavori, R.: Word sense disambiguation based on word sense clustering. In: Advances in Artificial Intelligence-IBERAMIA-SBIA 2006, pp. 472–481. Springer (2006)
2. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The berkeley framenet project. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1. pp. 86–90. Association for Computational Linguistics (1998)
3. Chow, I.C., Webster, J.J.: Mapping framenet and sumo with wordnet verb: Statistical distribution of lexical-ontological realization. In: Artificial Intelligence, 2006. MICAI'06. Fifth Mexican International Conference on. pp. 262–268. IEEE (2006)

4. Dannélls, D.: Applying semantic frame theory to automate natural language template generation from ontology statements. In: Proceedings of the 6th International Natural Language Generation Conference. pp. 179–183. Association for Computational Linguistics (2010)
5. Faure, D., Nédellec, C.: A corpus-based conceptual clustering method for verb frames and ontology acquisition. In: LREC workshop on adapting lexical and corpus resources to sublanguages and applications. vol. 707, p. 30 (1998)
6. Fillmore, C.J.: Scenes-and-frames semantics. Linguistic structures processing 59, 55–88 (1977)
7. Holmes, G., Donkin, A., Witten, I.H.: Weka: A machine learning workbench. In: Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on. pp. 357–361. IEEE (1994)
8. Johansson, R., Nugues, P.: A framenet-based semantic role labeler for swedish. In: Proceedings of the COLING/ACL on Main conference poster sessions. pp. 436–443. Association for Computational Linguistics (2006)
9. Katz, J.J., Fodor, J.A.: The structure of a semantic theory. language pp. 170–210 (1963)
10. Körner, S.J., Landhäußer, M.: Semantic enriching of natural language texts with automatic thematic role annotation. In: Natural Language Processing and Information Systems, pp. 92–99. Springer (2010)
11. Lakoff, G.: The invariance hypothesis: Is abstract reason based on image-schemas? Cognitive Linguistics (includes Cognitive Linguistic Bibliography) 1(1), 39–74 (1990)
12. Lenci, A., Busa, F., Ruimy, N., Gola, E., Monachini, M., Calzolari, N., Zampolli, A., Pustejovsky, J., Ogonowski, A., McCawley, C., et al.: Simple linguistic specifications. Deliverable D2 1 (2000)
13. Lenci, A., Montemagni, S., Venturi, G., Cutrulla, M.G.: Enriching the isst-tanl corpus with semantic frames. In: LREC. pp. 3719–3726 (2012)
14. Moreira, A., Salomão, M.M.M.: Applying bayesian networks and ontological types into lexeme to estimate the pertinence to a semantic frame. Revista Veredas 17(1), 149–164 (2013)
15. Moreira, A., Salomão, M.M.M.: Análise ontolológica aplicada ao desenvolvimento de frames. ALFA: Revista de Linguística 56(2) (2012)
16. Petruck, M.R.: Frame semantics. Handbook of pragmatics pp. 1–13 (1996)
17. Pradhan, S.S., Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., Weischedel, R.: Ontonotes: A unified relational semantic representation. International Journal of Semantic Computing 1(04), 405–419 (2007)
18. Pustejovsky, J.: The generative lexicon. Computational linguistics 17(4), 409–441 (1991)
19. Rayson, P.: From key words to key semantic domains. International Journal of Corpus Linguistics 13(4), 519–549 (2008)
20. Ruppenhofer, J., Ellsworth, M., Petruck, M.R., Johnson, C.R., Scheffczyk, J.: FrameNet II: Extended theory and practice (2006)
21. Sanfilippo, A., Tratz, S., Gregory, M., Chappell, A., Whitney, P., Posse, C., Paulson, P., Baddeley, B., Hohimer, R., White, A.: Ontological annotation with wordnet. In: Proceedings of the International WordNet Conference GWC (2006)

GODeM: A Graphical Ontology Design Methodology

Rafaela Blanca Silva-López[1], Mónica Silva-López[1], Maricela Bravo[1], Iris Iddaly Méndez-Gurrola[1], and Victor Germán Sánchez Arias[2]

[1] Universidad Autónoma Metropolitana, Distrito Federal, Mexico

[2] Universidad Nacional Autónoma de México, Distrito Federal, Mexico

{rbsl,misl,mcbc}@correo.azc.uam.mx, iddalym@yahoo.com.mx, victor_sanchez@cuaed.unam.mx

Abstract. In this paper we present a simple and didactic methodology to design an ontology for educational purposes. This methodology considers and incorporates the steps of the most outstanding methodologies for ontology design. Some of the reported methodologies specialize on the analysis of the knowledge domain, others in the formality of some of the language used to define it, others in the evaluation and documentation. Graphical Design Methodology (GODeM) is based on the methodological principles reported by Noy & McGuiness, the OntoDesign Methontology, Entreprise Ontology, Toronto Virtual Enterprise and graphical notations. GODeM methodology is used for designing an ontological model which main objective is to personalize learning activities consistent with the student's learning profile.

Keywords: Ontology design, ontology visualization, methodology for building ontologies, ontological model.

1 Introduction

Ontology design and construction is an arduous task which requires the organization of knowledge into standardized models, in order to categorize the information so it can be automatically processed by computers. The creation of intelligent systems requires ontological models, so it is necessary to have a simple and didactic methodology that facilitates the design and implementation of ontologies using a graphical notation that promotes the standardization of the graphical representation of ontology designs.

1.1 Conceptual Framework

The word ontology comes from the Greek roots ontos (being) and logos (treated). The German philosophers used to differentiate the study of being the study of the types of beings in the natural sciences.

The term ontology is adopted in Artificial Intelligence as a mechanism to share and reuse knowledge. Guarino defines the concept of ontology as a device constituted by a specific vocabulary that describes a knowledge domain, integrating a set of rules that specify such vocabulary [1][2]. While McGuinness defined ontology as the formal explicit description of concepts in a domain, including its properties and constraints that exist [3]. Although there are many different definitions of ontology, one of the most accepted is that of Thomas Gruber, who defined ontology as "a formal explicit specification of a shared conceptualization " [4].

Where conceptualization means that any ontology defines an abstract model (attributes, values and relationships) of the knowledge domain it represents. Explicit specification means that an ontology represents the description and representation of concepts in an unambiguous way. Formal, states that an ontology must be represented formally, so it can be reused, shared and understood by any agent or machine. Finally, the term shared concerns formal and explicit representation of concepts that have been agreed by a team of knowledge domain experts. It can be concluded that the main reason to build an ontology is to share information and reuse knowledge we have about a specific domain.

Based on the proposals of Gruber [4] and McGuinness [2], in this work, the term ontology refers to an explicit formal specification of concepts in a domain of shared knowledge, including their properties and constraints.

1.2 Components of an Ontology

From the point of view of engineering, an ontology is a device constituted by a specific vocabulary, used to describe a certain reality, includes a set of assumptions that determine the meaning of the vocabulary. Thus, components of an ontology is a hierarchy of classes with attributes and relationships, a semantic network that represents a set of interrelated instances, a set of axioms about classes and/or instances, and a set of rules inference. The literature shows that the components of a domain ontology depend on the interest and needs of the developer. They are based ontologies components proposed by Sowa, Noy & McGuinness and Farquhar [5-7].

2 Overview of Methodologies for Ontology Design

In this section we describe a set of related concepts concerned with methodologies for ontology design.

- Methodology: A set of methods and techniques that guarantee the quality of the results of an ontology design process.
- Method: ordered set of steps to develop a product.
- Technique: A procedure to achieve a goal [8]. Therefore, the methodology provides a framework to build an ontology for the domain of knowledge.
- Knowledge Engineering is the discipline derived from Artificial Intelligence responsible for the design and development of knowledge-based systems or expert

systems. It relies on instructional methodologies, ICT and Computer Science to represent knowledge in a domain of knowledge.

Ontology developers or engineers frequently search for a methodology to appropriately design an ontology; however, many variables are present and becomes a difficult task in many cases. There is currently no standard method for building ontologies, each methodology includes different steps and different considerations. Some methods used for the design of ontologies are listed in table 1.

Table 1**.** Ontology design methods.

Authors	Year	Methodology
University of Stanford Natalya F. Noy and Deborah L. McGuinness [6]	2000	Ontology Development 101: A Guide to Creating Your First Ontolog
Uschold and King [21]	1995	Enterprise Ontology
Grüninger and Fox [19]	1995	TOVE (Toronto Virtual Enterprise)
M. Uschold and M. Grüninger [20][22]	1996	ONTOLOGIES: Principles, Methods and Applications
Group of Ontological Engineering of the Polytechnic University of Madrid [12]	1997	Methontology

Methontology is a mature methodology, requires the integration of processes, in addition to requiring more detailed activities involved. Uschold, King, Grüninger, and Fox's methodologies do not describe activities, processes, techniques, or life cycle. Noy and McGuinness do not consider the documentation of the ontology.

2.1 Ontology Development Methodology

A methodology developed at Stanford University, proposed by Noy and McGuinness in [6]. It addresses the important aspects to be taken into account and suggests a method for ontology development. It proposes an iterative approach, adds details in each iteration, taking modeling decisions throughout the process. This methodology proposes the following steps for the design of ontologies: 1) Determine the domain and scope of the ontology [9]; 2) Consider reusing existing ontologies; 3) Enumerate important terms in the ontology; 4) Define the classes and their hierarchy [10]; 5) Define the properties and slots of classes; 6) Define the facets or restrictions on the properties and slots; and 7) Create instances.

This methodology focuses on understanding the knowledge domain for proper design of the ontology, describes the basic elements of the ontology, considers its implementation and validation to populate with data. It is the simplest methodology; however, it does not consider points such as the evaluation and documentation of the ontology.

2.2 Methontology

Methodology developed at the Polytechnic University of Madrid. Proposes an evolutionary prototyping process and procedure for the construction of an ontology [11-15]. Methonthology defines seven steps for the design of ontologies: 1) specification. [16]; 2) conceptualization; 3) acquisition; 4) integration; 5) implementation; 6) evaluation; and 7) documentation.

This methodology has a high degree of analysis to understand the domain of knowledge to model, constantly involves the knowledge domain experts, to collect information before the design of the ontology. We must be careful not to fall into an over- analysis which takes a long time. Methontology proposes the use of intermediate representations that facilitate the understanding of domain experts and formal languages. It has a strong foundation in knowledge engineering methodologies and software development process.

2.3 Enterprise Ontology Methodology

Enterprise Ontology is used as the basis of other proposed methods. The methodology includes four steps and provides design recommendations that should be present during the design and construction of the ontology [10], [17-18]: 1) Identify the purpose and scope of the ontology; 2) Building ontology (identify knowledge, encode knowledge and Integrate knowledge); 3) Evaluate the ontology; and 4) Document the ontology.

This methodology has only four steps, which greatly simplifies the work and proposes to encode the ontology in a formal language. It raises the need to evaluate the ontology through competency questions similarly as in software requirements specification. All methods start with identifying the purpose of the ontology and understanding of domain knowledge.

2.4 Gruninger and Fox Methodology

Toronto Virtual Enterprise (TOVE), this methodology proposes a scenario-based process to describe the functionality of the ontology [19-22]. The steps that found this methodology are six: 1) Identify relevant scenarios; 2) Develop relevant questions in an informal (natural language); 3) Specifying ontology terminology; 4) Develop relevant questions formally; 5) Specify the axioms and theorems; and 6) Evaluate the ontology. The key points of this methodology are: identify queries, objects and predicates in the ontology. Apply a high degree of formality as they resort to logical-mathematical language for the formal description of the relevant questions of the axioms and theorems.

Methontology is the most mature methodology; however, it does not consider competency questions, and the instantiation of individuals requires the incorporation of restrictions on properties. Grüninger and Fox's does not consider the reuse of existing ontologies. The metodologies of Uschold-King's and Grüninger-Fox's not describe activities, processes, techniques, or lifecycle. Finally, none of the

methodologies considers a graphical notation to represent the design of ontology clearly and simple way. It is therefore desirable to have a methodology that integrates these features.

3 Graphical Ontology Design Methodology (GODeM)

Guizzardi and Botti propose OntoUML in reprising the entity relationship model adapted to the modeling of ontologies, however, is unclear, and focuses on information representing each of the classes, its cardinality and relationship [23]. It is complex to represent all the features it has an ontology. Meanwhile, Ceccaroni and Kendall, propose a graphical environment for ontology development in which only make use of the UML class diagram, so there is no detail on the characteristics, properties and relationships of the various classes that make up the ontology [24].

Interactive Visualization of Large OWL Instance Sets, proposed by Liebig and Noppens, hierarchy diagrams used to represent the relationships between classes in the ontology, but does not include features, properties and detail of the relationships between classes. This type of diagram is informative, not graphically depicts all the features of an ontology [25].

Negru, Haag and Lohmann, have Unified Visual Notation for OWL Ontologies, which are used hierarchical diagrams using UML notation to represent the classes and their relationships to other classes. It does not include detailed information on the characteristics, properties and additional information relationships [26].

Furthermore, Lohmann, Negru, Haag, and Ertl, present the VOWL 2: User-Oriented Visualization of Ontologies, which allows graphically represent an ontology. This proposal is very similar to ours, using symbols to represent classes, properties, relationships, direction of relationships, cardinality, relationship types (data properties and object properties) and some colors to represent different types of properties [27].

OntoDesign Graphics can represent relationships between classes in the ontology as well as its characteristics and properties. Adds a graphical notation to integrate multiple ontologies and we only establish relationships between classes, but between ontologies. We use ovals instead of circles which gives more clarity to the graphical representation.

After analyzing the various methodologies for ontology design, we propose a methodology based on the principles of methodologies of Noy & McGuiness, Methontology, Grüninger Fox's, Enterprise Ontology and OntoDesign Graphics. The proposed methodology integrates the simplicity and detail offered by Noy & McGuiness methodology to understand the domain of knowledge and make a good design, at the same time, it integrates a graphical notation formal language that allows to visualize as a whole ontology design through OntoDesign Graphics. Finally incorporates the steppes of validation and documentation as required Methontology.

OntoDesign Graphics is a proposal for a notation that can represent grafically designing an ontology, visually in a single diagram can identify all the elements of the ontology, such as classes, class hierarchy, properties, relations between classes,

characteristics of properties, among others. Enrich documentation and facilitates the understanding of the design to other users [28].

OntoDesign Graphics integration having aim to have graphical notation that allows standardizing ontologies designs for clarity, as with the use of UML notation. For example, authors such as Rezgui, Mhiri, Ghédira, Ali,Tawil, Jahankhani, Yarandi, Mencke, Dumke, Bouhdidi, Ghailani, and Fennan [29-32], show a great diversity in the graphical representation of ontologies designs proposed which complicates interpretation between one notation and another.

We propose a methodology: Graphical Ontology Design Methodology (GODEM). The main goals of this methodology are: simplicity and didactic, used for teaching and educational. Facilitates the first ontology design a simple yet detailed guide you to achieve your goal.

GODeM the methodology is comprised of the following steps:

1. Specify the domain of knowledge and scope of the ontology.
   (a) Analyze the key elements involved in the domain of knowledge. Conduct interviews with experts in the domain of knowledge.
   (b) Prepare diagrams showing the relationships and characteristics of the key elements of the knowledge domain visually. Its aim is to facilitate feedback with expert domain knowledge.
2. Identification of requirements ontology.
   (a) Define the relevant questions that must be answered by ontology, also known as competency questions.
3. Validation of the possibility of using existing ontologies or metadata.
   (a) Browse and search in different repositories of ontologies related to the domain of knowledge that is addressed, to identify whether it is possible to reuse an ontology.
4. Ontological model design.
   (a) List important terms of ontology to develop a glossary of terms.
   (b) Define the classes and their hierarchy.
   (c) Define the properties or attributes of classes.
   (d) Define restrictions on properties (data type, cardinality, domain and range).
   (e) Elaborate design ontology with OntoDesign Graphics notation.
   (f) Populate the preliminar ontology design to detect and correct errors during design. In case of errors repeat the activities listed in subsection.
5. Implementation of the ontological model.
   (a) Select the language to use (OWL).
   (b) Select the tool for implementation (Protégé).
6. Populate classes.
   (a) Create instances or individuals populate the ontology with real world data.
7. Evaluation
   (a) Verification of ontology. Apply the rules established by [6]:
      (i) There are multiple solutions to model a domain. The best solution is given during the process depending on the purpose of the ontology and its applications.

(ii) The development of an ontology is an iterative incremental process.

(iii) Ontology classes are objects in the domain of knowledge and relationships are associated with verbs that are identified in the relevant questions that must be answered ontology.

(b) Validation of the ontology.

(i) Determine if the ontology answers the questions of competence.

8. Document the ontology.

(a) Document the steps taken during the design and implementation of ontology to share and reuse.

4 Case Study

The methodology used for the design of an educational ontological model in order to personalize learning activities to enhance learning and thus school passing rates. It is intended that the ontological model various cognitive theories applied to determine the student's learning profile, allowing you select learning activities that will improve their motivation and learning activities that promote the development of cognitive skills. Its aim is customizing learning activities from cognitive skills that develop students want.

To set the domain of knowledge and scope of the ontology, will discuss the key elements involved in the domain of knowledge and draw diagrams showing the relationships between the key elements and features appears. The diagrams facilitate communication with the domain expert knowledge, are a simple feedback and enabling understanding of the knowledge domain in question.

4.1 Analysis of the Key Elements Involved in the Domain of Knowledge

The key elements involved in the domain of knowledge are: personalization learning activities, the learning profile of the student, the course, and the student's general data.

To identify the requirements of the ontology is necessary to develop a list of relevant questions that must be answered by the ontology.

For the domain of knowledge we found the following competency questions:

What is the domain of the ontology? The ontology focuses on the educational domain, specifically in the teaching-learning process. Particularly focusing on the customization of the assessment.

What is the use of the ontology? To customize assessment activities in accordance with the profile of an individual's learning and mastering knowledge of teaching a course. The experimental evaluation case that will be used is the course "Structured Programming" with engineering students.

Who will use the ontology? Ontology users are students and teachers. For the particular case of fieldwork, students will apply Engineering with massive semi - face courses, the Structured Programming course.

4.2 Competency Questions

The following list is the set of questions that the ontology should answer:

1. What are the cognitive types for a cognitive theory X?
2. What characteristics does a guy cognitive Z for a cognitive theory Y?
3. What is the student's learning profile X?
4. What are the cognitive characteristics that a student X has?
5. What learning activities are recommended for the course that requires developing the cognitive ability Y?
6. What learning activities are recommended for learning profile X?
7. What kind of tool should be recommended to perform an activity that develops a cognitive skill Y?
8. What assessment activity should make a student with learning profile for the course W X Y module?
9. What is the recommended learning path for a student with learning profile W?

4.3 Identification of Key Concepts and Axiomatization of the Ontology

The development of the conceptual model of ontology starts with the list of key terms that relate to the field of knowledge that is addressed and worked the glossary of terms shown in table 2. The axiomatization establishes necessary and sufficient restrictions for class properties. It is important to add annotations to the classes for a formal design documentation. Now, to axiomatize must define constraints on the properties, therefore specifies the data type, type of cardinality, as well as its domain and range. The axiomatization of classes of Profiles ontology is shown in table 3.

Table 2. Glossary of terms.

Concept	Description
Student	Individual requires a personalized learning path.
Learning profile	Characteristics that differentiate people and for determining how to learn and think.
Course	Thematic content of the discipline to be taught.
Evaluation	Mechanisms to verify the knowledge acquired by the student.
Module	Section of the course addresses a specific topic as part of the course.
Multimedia Educational Resource	Multimedia educational material oriented to student learning. Incluiding videos, recorded lectures, electronic books, among others.
Cognitive Ability	Skills to be developed by the student to complete course.
Evaluation Type	Characteristics that determine the type assessment student knowledge.
Activity	Learning activities performed by the student.
Cognitive Style	Determine the characteristics that identify a learning profile.
Learning Path	The system offers to personalize the activities of student per module.

Table 3. Axiomatization of classes of Profiles ontology.

Class	Property	Data type	Cardinality
CognitiveStyle	Description	String	≡ 1
	Author	Symbol	≡ 1
	URLTest	String	$0 \geq \leq 1$
CognitiveType	Name	String	≡ 1
CognitiveFeature	Description	String	≡ 1

4.4 Design and Implementation of the Ontology

The design of the ontological model consists of 5 ontologies: Profiles, Students, Courses, AssessmentActivities and LearningPath. This ontological model stores the profile of student learning, and from cognitive skills identified, the activities are customized in order to foster the development of skills in line with the objectives of each course unit. OWL DL is the standard description formal language to specify ontologies, the reasoning is Pellet and the tool used to implement the ontology is Protégé as shown in figure 1. The ontology was modeled with OntoDesign Graphics, and it's shown in the figure 2.

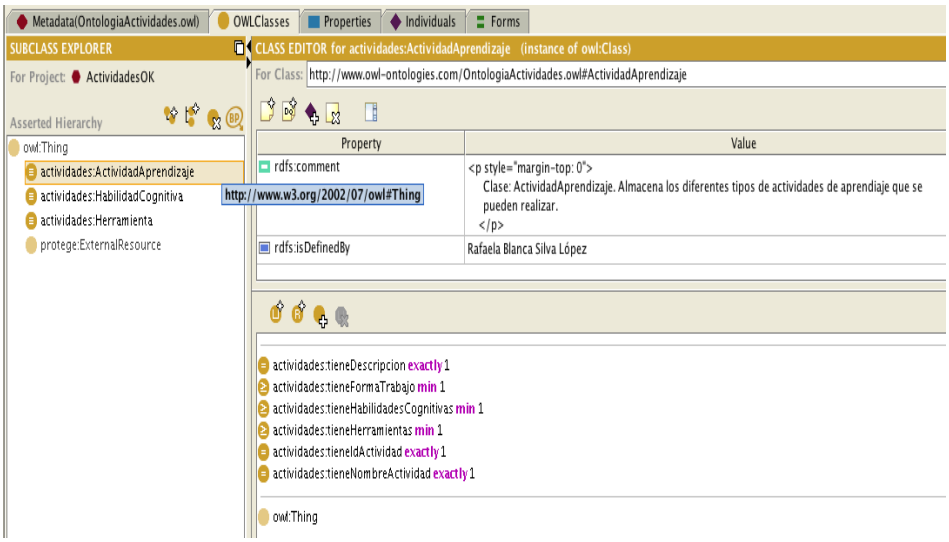

Fig. 1. Implementing the ontology in Protégé.

5 Conclusions

In this paper, we have described the Graphical Ontology Design Methodology (GODeM). One of the most important things is the incorporation of use OntoDesign Graphics notation.

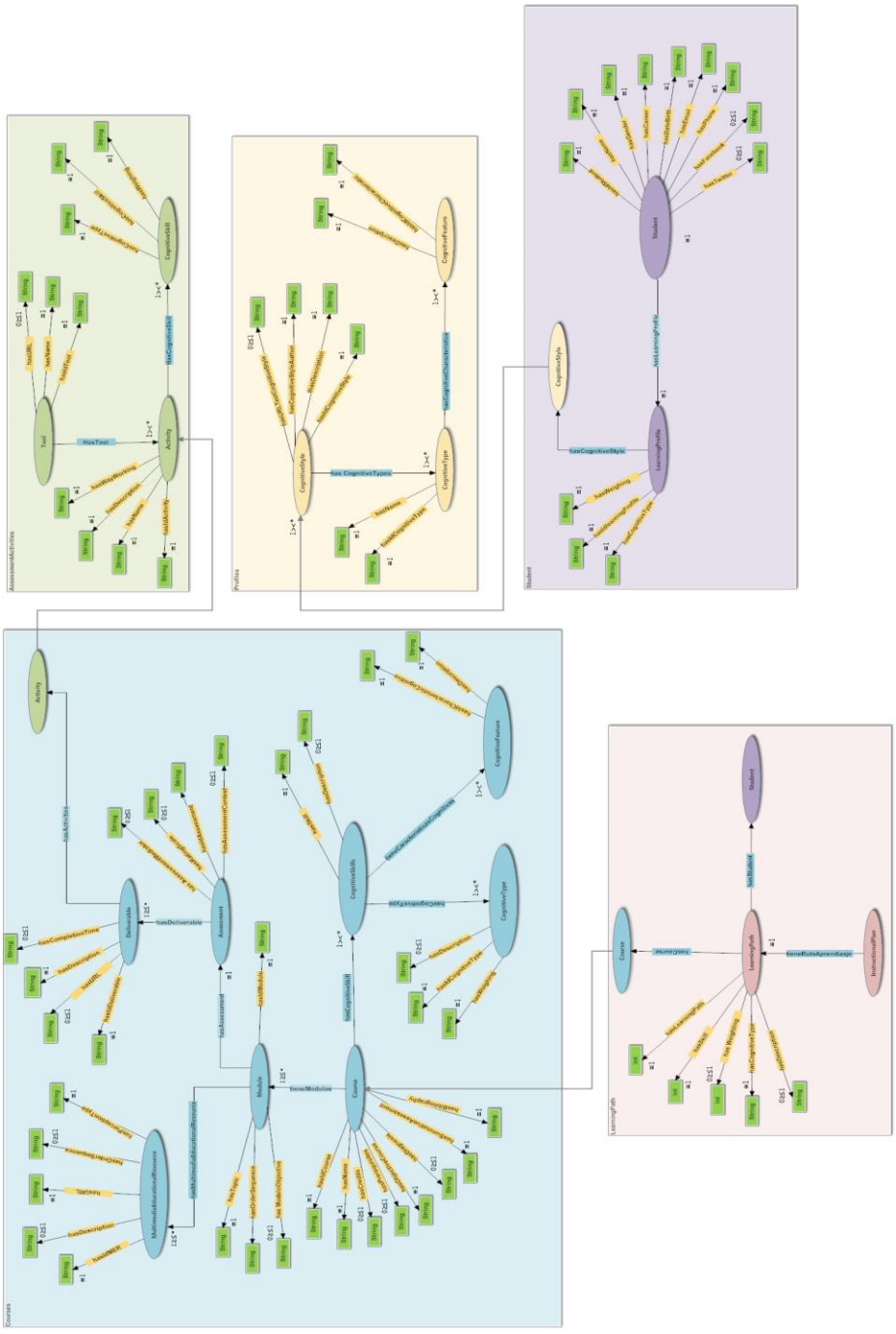

Fig. 2. Semantic relations between ontologies with OntoDesign Graphics.

However, the ontology design is a creative processand, two ontologies designed by different people would be different. The potential applications of the GODeM is incorporating a graphical notation used in the design of ontologies to have an easier to reuse documentation. Finally, we can assess the quality of our ontology by using it in applications for which we designed it. The results generated determine adjustments that must be made.

Acknowledgements. This work is part of the research undertaken by Blanca Silva to obtain the PhD at UDG-Virtual, México, and it is supported by Universidad Autónoma Metropolitana. Also this work is part of the project PAPIIT UNAM IT100213.

References

1. Guarino, N.: Understanding, Building and Using Ontologies. International Journal of Human Computer Studies. pp. 293–310 (1997)
2. Guarino, N.: Formal Ontology and Information Systems. Proceedings of the 1st International Conference on Formal Ontologies in Information Systems, FOIS'98, pp. 3–15. IOS Press (1998)
3. McGuinness, D.L: Ontologies Come of Age. Fensel, D., Hendler, J., Lieberman, H., Wahlster, W. Editors. The Semantic Web: Why, What, and How, MIT Press, (2003)
4. Gruber,T.R.: A Translation Approach to Portable Ontologies. Knowledge Acquisition, pp. 199–220 (1993)
5. Sowa, J.F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Pacific Grove, CA: Brooks Cole Publishing Co (2000)
6. Noy, N.F., McGuinness, D.L.: Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880 (2001)
7. Farquhar, A.: Ontolingua Tutorial. Knowledge Systems Lab. University of Stanford. [Online].
8. Ander-Egg, E.: Técnicas de investigación social para trabajadores sociales. Buenos Aires: El Cid Editor (1978)
9. Grüninger, M., Fox, M.S.: Methodology for the Design and Evaluation of Ontologies. In Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95 (1995)
10. Uschold, U., Grüninger, M.: Ontologies: Principles, Methods and Applications. Knowledge Engineering. Review, pp. 93–155 (1996)
11. Gómez-Pérez, A.: A Framework to Verify Knowledge Sharing Technology. Expert Systems with Application, pp. 519–529 (1996)
12. Fernández, M., Gómez-Pérez, A., Juristo, N.: METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In Proceedings of AAAI97 Spring Symposium Series, Workshop on Ontological Engineering, pp. 33–40 (1997)
13. Gómez-Pérez, A.: Knowledge Sharing and Reuse. In J. Liebowitz (Ed.) Handbook of Expert Systems. CRC (1998)
14. Fernández, M.: Overview of Methodologies for Building Ontologies. In V. R Benjamins (Ed.) Proceedings of IJCAI99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends, Vol. 18, CEUR Publications (1999)

15. Fernández, M., Gómez-Pérez, A., Pazos-Sierra, A., Pazos-Sierra, J.: Building a Chemical Ontology Using Methontology and the Ontology Design Environment. IEEE Intelligent Systems, pp. 37–46 (1999)
16. García Peñalvo, F.J.: Web Semántica y Ontologías. (2005) [Online]. Available: http://zarza.usal.es/~fgarcia/doctorado/iuce/WSemantica.pdf
17. Uschold, M., King, M.: Towards a Methodology for Building Ontologies. In Proceedings of IJCAI95's Workshop on Basic Ontological Issues in Knowledge Sharing (1995)
18. Uschold, M.: Building Ontologies: Towards a Unified Methodology. In Proceedings of 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems (1996)
19. Grüninger, M., Fox, M.S.: The Design and Evaluation of Ontologies for Enterprise Engineering. In Proceedings of the Workshop on Implemented Ontologies, European Conference on Artificial Intelligence (1994)
20. Grüninger, M., Fox, M.S. The Role of Competency Questions in Enterprise Engineering. In Proceedings of the IFIP WG5.7 Workshop on Benchmarking–Theory and Practice. (1994)
21. Grüninger, M., Fox, M.S.: The Logic of Enterprise Modelling. In Brown, J., O'Sullivan, D. (Eds.), Reengineering the Enterprise, pp. 83–98, Chapman and Hall (1995)
22. Grüninger, M.: Designing and Evaluating Generic Ontologies. In Proceedings of the 12th European Conference of Artificial Intelligence, pp. 53–65 (1996)
23. Botti, A., Guizzardi, G.: A Model-Based Tool for Conceptual Modeling and Domain Ontology Engineering in OntoUML, Springer-Verlag Berlin Heidelberg, pp. 528–539 (2009)
24. Ceccaroni, L., Kendall, E.: A Graphical Environment for Ontology Development. ACM 1-58113-683-8/03/0007, pp. 958–959 (2003)
25. Noppens, O., Liebig, T.: Interactive Visualization of Large OWL Instance Sets. In Proceedings of the Third Int. Semantic Web User Interaction Workshop, 2006. Athens, GA, USA (2006)
26. Negru, S., Haag, F., Lohmann, S.: Towards a Unified Visual Notation for OWL Ontologies: Insights from a Comparative User Study. In Proceedings of the 9th International Conference on Semantic Systems, New York, NY, USA: ACM (2013)
27. Lohmann, S., Negru, S., Haag, F., Ertl. T.: VOWL 2: User-Oriented Visualization of Ontologies. EKAW, (2014)
28. Silva-López, R.B., Silva-López, M., Méndez-Gurrola, I.I., Bravo, M.: Onto Design Graphics (ODG): A Graphical Notation to Standardize Ontology, MICAI 2014, Part I, LNAI 8856, Springer International Publishing Switzerland, pp. 443–452 (2014)
29. Rezgui, K., Mhiri, H., Ghédira, K.: An Ontology-based Profile for Learner Representation in Learning Networks. International Journal of Emerging Technologies in Learning, Vol. 9 (3), pp. 16–25 (2014)
30. Seyed Ali, H., Abdel-Rahman, H.T, Jahankhani, H., Yarandi, M.: Towards an Ontological Learners' Modelling Approach for Personalised e-Learning. In International Journal of Emerging Technologies in Learning, Vol. 8(2), pp. 4–10 (2013)
31. El Bouhdidi, J., Ghailani, M., Fennan, A.: A Probabilistic Approach for the Generation of Learning Sessions Tailored to the Learning Styles of Learners. In International Journal of Emerging Technologies in Learning, Vol. 8(6), pp. 42–49 (2013)
32. Mencke, S., Dumke, R.: Didactical Ontologies. In International Journal of Emerging Technologies in Learning, Vol. 3(1), pp. 65–73 (2008)

Text Recognition with k-means Clustering

Mohammad Iman Jamnejad, Ali Heidarzadegan, and Mohsen Meshki

Department of Computer Engineering, Beyza Branch, Islamic Azad University, Beyza, Iran

jamnejad@beyzaiau.ac.ir

Abstract. A thesaurus is a reference work that lists words grouped together according to similarity of meaning (containing synonyms and sometimes antonyms), in contrast to a dictionary, which contains definitions and pronunciations. This paper proposes an innovative approach to improve the classification performance of Persian texts considering a very large thesaurus. The paper proposes a flexible method to recognize and categorize the Persian texts employing a thesaurus as a helpful knowledge. In the corpus, when utilizing the thesaurus the method obtains a more representative set of word-frequencies comparing to those obtained when the method disables the thesaurus. Two types of word relationships are considered in our used thesaurus. This is the first attempt to use a Persian thesaurus in the field of Persian information retrieval. The k-nearest neighbor classifier, decision tree classifier and k-means clustering algorithm are employed as classifier over the frequency based features. Experimental results indicate enabling thesaurus causes the method significantly outperforms in text classification and clustering.

Keywords: Persian texts, Persian thesaurus, semantic-based text classification, k-nearest neighbor.

1 Introduction

Nowadays, usage of recognition systems has found many applications in almost all fields [23-35]. K-Nearest Neighbor (kNN) classifier is one of the most fundamental recognition systems. It is also the simplest classifier. It could be the first choice for a classification study when there is little or no prior knowledge about the data distribution. It has been shown that it is effective in many fields such as text categorization field [36-37], intrusion detection filed [38] (that is first converted text categorization problem then treats it as text categorization), medical systems such as diagnosis of diabetes diseases [39], thyroid diseases [40] and myocardial infarction [41], and image classification [42] and etc. It has been shown that kNN is a successful classifier for text categorization [36-38].

Clustering is the assignment of objects into groups (called clusters) so that objects from the same cluster are more similar to each other than objects from different

clusters [25], [28] and [32]. In data mining, k-means clustering is a method of cluster analysis which aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean. Although k-means is considered as a clustering algorithm, in this paper it is employed as a classifier, it means we assume that labels are given in its evaluation, for comparing with kNN classifier. While it has been shown that employing thesaurus can improve the text clustering and classification in Latin languages [43-44], we also aim that evaluate whether employing Persian thesaurus improves text clustering or not.

Decision Tree (DT) is considered as one of the most versatile classifiers in the machine learning field. DT is considered as one of unstable classifiers. It means that it can converge to different solutions in successive trainings on same dataset with same initializations. It uses a tree-like graph or model of decisions. The kind of its knowledge representation is appropriate for experts to understand what it does [45].

In the current century Information Technology is considered as one of the most important scientific fields (if not the most important field) among the researchers. Ever-increasing growth pace of data makes its appropriate and efficient management significantly important and also its appropriate usage inevitable. Indeed proper responding to user queries is considered as a crucial challenge in the Information Technology [1]. Two of the most important challenging problems in the field of Information Technology include:

- How can one handle information retrieval problem in a huge number of texts efficiently?
- How can one extract useful information out of a huge mass of data efficiently?

From this perspective, usage of text keywords has been considered as a very promising approach for researchers to handle two mentioned challenges.

A thesaurus is a reference work that lists words grouped together according to similarity of meaning (containing synonyms and sometimes antonyms), in contrast to a dictionary, which contains definitions and pronunciations. This paper proposes to use existing between-word-relationships to help us build an automatic thesaurus-based indexing approach in Persian language.

2 Related Works

In 1999, Turney showed that keyword extraction field is one of the most important factors accelerating and facilitating the information retrieval applications, but until then there is no attempt to improve the quality of extracted keywords [5].

Simultaneously in 1999, Frank et al., who worked in the field of artificial intelligence, tried to improve the quality of extracted keywords by presenting machine processing algorithm. Their work was based on a Simple Bayes algorithm. Their system is named "KEA". In the KEA method, although the quality of extracted keywords significantly increased, linguistic issues were not taken into considerations during keyword extraction process [6]. The general process of keyword extraction was introduced by Liu et al. in 2005. They first elect a number of candidate words as

potential keywords, then assign a weight to each potential keyword, and finally consider potential keywords with the highest weights as the final extracted keywords [7]. Franz in 2002 combined statistical analysis and linguistic analysis [8]. He believed that without considering information about linguistic knowledge, statistical analysis considers disadvantageous and non-keywords [8].

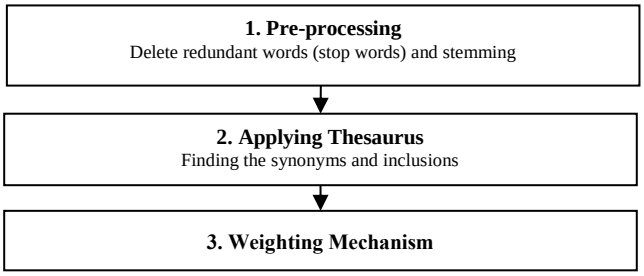

Fig. 1. Proposed indexing framework.

Along with previous researches, to solve drawbacks of the traditional keyword extraction approaches (that extract disadvantageous and non-key words instead of the keywords), Freitas et al. modeled process of the keyword extraction into a classification problem in 2005 [9]. Zhang et al. used a decision tree as classifier to recognize the keywords among all words [10]. Halt used the features based on N-gram concept in the context of information retrieval [11]. In the first attempt, Deegan used thesaurus concept in 2004 to improve information retrieval efficacy [12]. After that Hyun tried to use a specialized thesaurus for special-formatted queries [13]. There are some successive works that try to improve information retrieval efficacy after then [14]-[16].

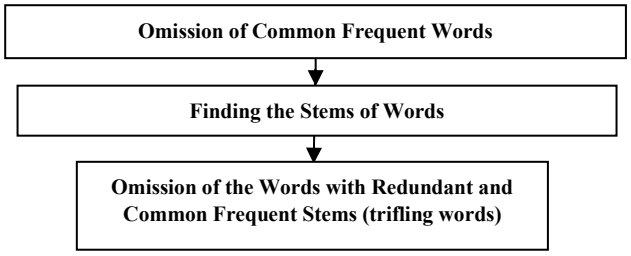

Fig. 2. Pre-processing phase of proposed framework.

There are some related works done in the field of Persian language. While there are many methods in Persian language, there is a lack of employing a thesaurus in Persian so far. The curious reader is referred to [4] and [17]-[20] for more detail. The only work that employs a thesaurus is Parvin et al. work that is a very simple and immature one [21].

3 Proposed Framework

Fig. 1 depicts the proposed framework. The first step in Fig. 1 is expanded in Fig. 2. As seen in Fig. 2, in preprocessing step, Persian texts are refined into useful texts to get rid of the trivial words that are unnecessary for keyword extraction phase.

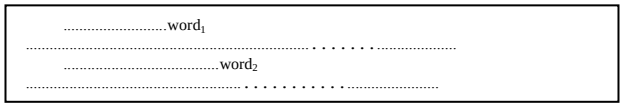

Fig. 3. A typical text with three words that are synonyms.

Indeed the pre-processing step of proposed framework consists of three phases (sub-steps). In first phase the common frequent words like prepositions are omitted. Then the stem of each word is found. Third the common frequent stems, like "*be*", are also omitted from the text.

Table 1. Table with frequencies of words of Fig. 3.

word	frequency	Type
.	.	.
.	.	.
.	.	.
word1	3	Head
word2	3	Child
word3	3	Child
.	.	.
.	.	.
.	.	.

To clarify second step, please consider Fig. 3. In Fig. 3 assume that the $word_1$, $word_2$ and $word_3$ are synonyms of each other. Using a thesaurus these three words, i.e. $word_1$ and $word_2$ and $word_3$ are considered as the single word that is first observed, i.e. $word_1$ with a frequency as many as sum of their frequencies, here 3. Here $word_1$ is head word of those three words and two words, $word_2$ and $word_3$, are children of head word $word_1$. So after second step a table of words is obtained from the input text that depicts the words next to their frequencies; for example the table of words for the text presented in Fig. 3 is like Table 1.

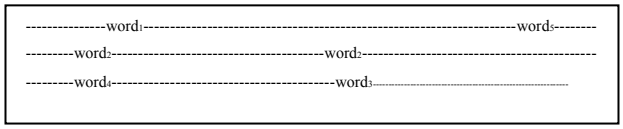

Fig. 4. A typical text with five words.

In the table of words, words are partitioned into two types: (a) *head* type and (b) *child* type. Only words with *head* type are considered in the final step. Consider the

table of words extracted from the previous example and presented in Table 1. It contains three words, $word_1$, $word_2$ and $word_3$. Only the word $word_1$ is considered as *head* type and its frequency is equal to 3. Two other words are considered as *child* type.

So in obtaining a table of words, weight for a synonym/antonym relationship is considered by a one, i.e. each occurrence the synonym/antonym of a word is equal to an occurrence the original word. Another relationship that is taken into consideration is inclusion. For example a word like *animal* includes a *wolf*. So in a text that has a word *animal* as a head type word, occurring a word *wolf* is equal to occurring a word *wolf* and also occurring the *head* type word *animal* with weight α, where α is less than one and vice versa. It means if an inclusion word has been occurred as a *head* type word so far, occurring an included word is to occur the included word by weight one, and including word by a weight α, where α is a real number below one. For example consider text presented in Fig. 4. Assume that $word_5$ is a special kind of $word_4$ and $word_4$ is a special kind of $word_3$. As before, $word_1$, $word_2$ and $word_3$ are synonyms/antonym of each other.

Now a table is extracted from the text presented in Fig. 4 that the frequencies of its words are like Table 2. For simplicity we assume that α is 1/4 for this example.

In Table 2, word $word_1$ is the *head* for three words, $word_1$, $word_2$ and $word_3$. Because those words, $word_1$, $word_2$ and $word_3$, are occurred 4 times, their frequencies are considered 4 at least. Besides, due to occurring the word $word_4$ that is a special kind of $word_3$, a 1/4 (α) is added to their frequencies. Due to occurring the word $word_5$ that is a special kind of word $word_4$, a 1/4*1/4 (α^2) is added to their frequencies. From another side, the frequency of the word $word_4$ is at least 1, due to its one direct appearance. Because of four appearances of the word $word_1$, 4 times 1/4 (4*α) is added by its one appearance. Besides because of one appearance of word $word_5$ another 1/4 (α) is added to its frequency. This scenario is valid for the word $word_5$. It means that one appearance of the word $word_5$, plus 1/4 (α) due to appearance of the word $word_4$ plus 4 appearances of the word $word_1$ that has inclusion relationship with length 2, i.e. 4×1/4×1/4 ($4\times\alpha^2$), is considered as frequency of the word $word_5$.

Table 2. Table with frequencies of words of Fig. 4.

word	frequency	type
.	.	.
.	.	.
.	.	.
$word_1$	4+1/4+1/4*1/4	$head_i$
$word_2$	4+1/4+1/4*1/4	$child_i$
$word_3$	4+1/4+1/4*1/4	$child_i$
$word_4$	1+4*1/4+1/4	$head_{i+1}$
$word_5$	1+1/4+4*1/4*1/4	$head_{i+2}$
.	.	.
.	.	.
.	.	.

4 Experimental Studies

Employed criteria based on which an output of a classifier or a clustering algorithm are evaluated, are discussed in the first part of this section. The details of the used dataset are given in the subsequent part. Then the settings of experimentations are given. Finally the experimental results are presented.

We have two different parts of experimentations. In the first part of experimentations we use a simple classifier to show the effectiveness of the proposed method. We employ confusion matrix to visually show the distribution of articles in different classes. Each row in the confusion matrix represents the instances in a predicted class, while each column of the confusion matrix represents the instances in an actual class. One benefit of a confusion matrix is that it is easy to see if the system is confusing two classes. To evaluate the performance of the classification, the accuracy, entropy and purity measures are taken as the evaluation metrics throughout all the paper. Accuracy is computed according to equation 1:

$$Acc\ (L) = \frac{\sum_{i=1}^{k_n} n_{ii}}{n}, \tag{1}$$

where n is the total number of samples and n_{ij} denotes the number patterns of class j that are classified by classifier L as class i. Consider a discrete random variable X, with N possible values $\{x_1, x_2, ..., x_N\}$ with probabilities $\{p(x_1), p(x_2), ..., p(x_N)\}$. Entropy of discrete random variable X is obtained using equation 2.

$$E(X) = -\sum_{i=1}^{N} p(x_i) \log\ p(x_i). \tag{2}$$

And its purity is obtained using equation 3.

$$P(X) = \max\ p(x_i). \tag{3}$$

We can consider i-th row of the confusion matrix as a distribution of patterns in the class i and evaluate the purity and entropy measures for the class. Then by considering a weight n_i/n for class i, where n_i is number of the samples in class i and n is total samples, we sum the purities and entropies of all classes. It means for classifier L the purity and entropy measures are computed as equations 4 and 5 respectively.

$$E(L) = \sum_{i=1}^{c} \frac{n_i}{n} * E(c_i), \tag{4}$$

where c is number of classes and $E(c_i)$ is the entropy of class i.

$$P(L) = \sum_{i=1}^{c} \frac{n_i}{n} * P(c_i). \tag{5}$$

All the classification experiments are done using 4-fold cross validation. The results obtained by 4-fold cross validation are repeated as many as 10 independent

runs. The averaged accuracies over the 10 independent runs are reported. Confusion matrix of 1-nearest neighbour classifier with leave-one-out technique is presented as a comprehensive study of performance of classification.

In the second part of experimentations k-means clustering algorithm is applied over dataset. Here the normalized mutual information (NMI) between the output partition and the real labels of dataset is considered as the main evaluation metric of the final partition [2]. The NMI between two partitionings, P^a and P^b, is calculated based on equation 6.

$$NMI\ (P^a, P^b) = \frac{-2\sum_{i=1}^{k_a}\sum_{j=1}^{k_b} n_{ij}^{ab} \log\left(\frac{n_{ij}^{ab}.n}{n_i^a.n_j^b}\right)}{\sum_{i=1}^{k_a} n_i^a \log\left(\frac{n_i^a}{n}\right) + \sum_{j=1}^{k_b} n_j^b \log\left(\frac{n_j^b}{n}\right)}, \quad (6)$$

where n is the total number of samples and n_{ij}^{ab} denotes the number of the shared patterns between clusters $C_i^a \in P^a$ and $C_j^b \in P^b$; n_i^a is the number of the patterns in cluster i of partition a; also n_j^b is the number of the patterns in cluster j of partition b.

Second alternative to evaluate a partition is the accuracy metric, provided that the number of clusters and their true assignments are known. To compute the final performance of k-means clustering in terms of accuracy, one can first re-label the obtained clusters in such a way that have maximal matching with the ground true labels and then counting the percentage of the true classified samples. So the error rate can be determined after solving the correspondence problem between the labels of derived and known clusters. The Hungarian algorithm is employed to solve the minimal weight bipartite matching problem. It has been shown that it can efficiently solve this label correspondence problem [46].

Table 3. Details of used dataset.

Row	Topic	# of articles	Average # of words	Average # of words after refinement phase
1	Sport	146	204	149
2	Economic	154	199	135
3	Rural	171	123	76
4	Adventure	89	160	115
5	Foreign	130	177	124

In order to test the proposed method five different categories have been collected from Hamshahri newspaper [3]. The detail of the dataset is presented in the Table 3.

After refinement of dataset, the average number of words in each category is reduced as the last column of Table 3. And then after applying refinement phase, we produce a feature space as illustrated in Table 4.

In Table 4, parameter n is the number of the words which are considered as *head* word type in one article at least. The entity j-th column of i-th row in Table 4 is equal to frequency value of *head* word j in i-th article. The parameter m that shows the

number of articles in dataset is 400. It means 75 articles per class. The averaged number of features in dataset, *n*, is 171.5.

Table 4. Dataset after refinement.

	$Head\ Word_1$	$Head\ Word_2$	$Head\ Word_3$	.....	$Head\ Word_n$
$Article_1$					
$Article_2$					
…					
$Article_m$					

Table 5. Performances of 1-NN classifier and k-means clustering with and without thesaurus.

	Without thesaurus	With thesaurus
1-NN Accuracy	70.49%	81.16%
1-NN Entropy Measure	0.95	0.69
1-NN Purity Measure	70.49%	81.16%
1-NN F-Measure	70.76%	81.43%
1-NN NMI	20.08%	28.20%
DT Accuracy	67.57%	82.03%
DT Purity Measure	69.08%	81.43%
k-means Accuracy	64.78%	72.61%
k-means Entropy Measure	1.06	0.91
k-means Purity Measure	64.78%	72.61%
k-means F-Measure	65.14%	72.83%
k-means NMI	16.65%	21.38%

Table 6. Confusion between Class-Cluster in Document example employing thesaurus.

Cluster	Sport	Economic	Rural	Adventure	Foreign	Entropy	Purity
1	2	1	2	123	4	0.34	93.18
2	13	120	3	6	4	0.69	82.19
3	132	14	5	2	2	0.58	85.16
4	19	7	73	11	8	1.16	61.86
5	5	12	6	4	112	0.74	80.58
Total	171	154	89	146	130	0.69	81.16

By filling values of Table 4 by using thesaurus and without using thesaurus we obtain two different datasets. Thesaurus is a collection of words, phrases and information about a specific field of human wisdom. This collection of words is organized to integrate and centralize vocabulary in the field to make it easy understand the relation between the previous concepts. The used thesaurus is produced considering the manual presented by Hori [22].

Table 7. Confusion between Class-Cluster in Document example without employing thesaurus.

Cluster	Sport	Economic	Rural	Adventure	Foreign	Entropy	Purity
1	17	99	4	11	9	0.98	70.71
2	119	24	7	2	3	0.76	76.77
3	9	18	7	8	94	1.02	69.12
4	3	2	7	109	15	0.72	80.15
5	23	11	64	14	9	1.31	52.89
Total	171	154	89	144	130	0.95	70.49

We use 1-neareast neighbour classifier as base classifier and averaged on 10 independent runs each of which obtained by 4-fold cross validation is reported. Parameter α is considered 1/4 throughout all the experimentations. The true labels of this dataset are employed for obtaining the accuracy metric. For reaching the matrices (Table 6 and Table 7) we use 1-neareast neighbour and leave-one-out technique. In clustering the real number of cluster (here 5) is feed to k-means algorithm. The similarity measure to reach similarity matrices is based on normalized Euclidean distance.

Table 5 shows the main results of first part of experimentations. The table shows in first row Accuracy measures of 1-NN (nearest neighbour) classifier with and without thesaurus. It then shows the Entropy and Purity measures of the classifier in the two subsequent rows. Then it presents k-means clustering accuracy and NMI measures in the two subsequent rows. The matrix representing confusion between class-cluster in document example for 1-NN classifier on features obtained by help of thesaurus is shown in the Table 6. The entropy for each cluster is calculated based on equation

$$entropy_j = \sum_{i=1}^{c} p_{ij} \log_2(p_{ij}),$$

where c is the number of classes, and p_{ij} is m_{ji}/m_j. m_{ji} is the number of instances of class i in cluster j, and m_j is the number of instances in cluster j. The purity is calculated based on equation

$$purity_j = max_{i=1}^{c}(p_{ij}).$$

The total purity is

$$purity_j = \sum_{j=1}^{q} {m_j}/{m} \times purity_j ,$$

where q is the number of clusters, and m is the number of total instances. The total entropy is

$$entropy_j = \sum_{j=1}^{q} {m_j}/{m} \times entropy_j .$$

Accuracy is calculated based on equation

$$Accuracy_j = \sum_{j=1}^{q} m_j \times purity_j / m.$$

The same matrix for 1-NN classifier on features obtained without help of thesaurus is shown in the Table 7.

5 Conclusion and Future Works

In this paper, we have proposed a new method to improve the performance of Persian text classification. The proposed method uses a Persian thesaurus to reinforce the frequencies of words. With a simple classifier, it is shown that using thesaurus can improve the classification of Persian texts. We consider two relationships: synonyms and inclusion. We use a hierarchical inclusion weighting, and linear synonym weighting. As it is concluded the text classification and clustering both can be significantly improved in the case of applying a thesaurus.

As a future work, one can turn to research on the different weighting methods. For another further future work it can be studied how further relationships, like contradiction, can affect the text classification performance.

References

1 American Society of Indexers. Frequently Asked Questions Indexing. Index review in Books, Ireland. Available: http://www.asindexing.org/site/indfaq.shtml
2 Strehl A. and Ghosh J.: Cluster ensembles - a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research, 3:583–617 (2002)
3 Hamshahri newspaper, http://www.hamshahrionline.ir
4 Yousefi, A.: Principles and methods for computerized indexing. Journal Books. Volume 9, Number 2 (2010) (in Persian)
5 Turney, P.D.: Learning Algorithms for Keyphrase Extraction. Information Retrieval, 2(4), pp. 306–336 (1999)
6 Frank, E.: Domain-Based Extraction of Technical Keyphrases. In: International Joint Conference on Artificial Intelligence, India (1999)
7 Liu, Y., Ciliax, B.J., Borges, K., Dasigi, V., Ram, A., Navathe, S.B., Ingledine, R.: Comparison of two schemes for automatic keyword extraction from MEDLINE for functional gene clustering. Computational Systems Bioinformatics Conference, Stanford (2005)
8 Frantzi, K., Ananiadou, S., Mima, H.: Automatic Recognition of Multi-word Terms: the C-value/NC-value Method. Digital Libraries, 3(2), pp. 115–130 (2002)
9 Freitas, N., Kaestner, A.: Automatic text summarization using a machine learning approach. In: Brazilian Symposium on Artificial Intelligence (SBIA), Brazil (2005)
10 Zhang, Y., Heywood, N.Z., Milios, E.: World Wide Web Site Summarization Web Intelligence and Agent Systems. Technical Report, CS-2002-8 (2006)
11 Hult, A.: Improved automatic keyword extraction given more linguistic knowledge. In: 8th Conference on Empirical Methods in Natural Language Processing (2003)
12 Deegan, M.: Keyword Extraction with Thesauri and Content Analysis. URL: http://www.rlg.org/en/page.php?Page_ID=17068

13 Hyun, D.: Automatic Keyword Extraction Using Category Correlation of Data. Heidelberg, pp. 224–230 (2006)
14 Witten, W., Medley, I.H.: Thesaurus based automatic keyphrase indexing. In: 6th ACM/IEEE-CS JCDL '06 (Joint Conference on Digital Libraries) (2006)
15 Klein, M., Steenbergen, W.V.: Thesaurus-based Retrieval of Case Law. In: 19th International JURIX conference, Paris (2006)
16 Martinez, J.L.: Automatic Keyword Extraction for News Finder. Heidelberg, pp. 405–427 (2008)
17 Shahabi, A.M.: Abstract construction in Persian literature. In: Second International Conference on Cognitive Science, p. 56, Tehran (2002) (in Persian)
18 Bahar, M.T.: Persian Grammar. Chapter IV, p. 111 (1962) (in Persian)
19 Khalouei, M.: Indexing machine. Journal Books, Volume 6, Number 3 (2009) (in Persian)
20 Karimi, Z., Shamsfard, M.: Automatic summarization systems Persian literature. In: 12th International Conference of Computer Society of Iran (2005) (in Persian)
21 Parvin, H., Minaei-Bidgoli, B., Dahbashi, A.: Improving Persian Text Classification Using Persian Thesaurus. In: Iberoamerican Congress on Pattern Recognition, pp. 391–398 (2011)
22 Hori, E.: A manual to make and develop a multilingual thesaurus. Scientific Documentation Center (2003) (in Persian)
23. Daryabari M., Minaei-Bidgoli B., Parvin H.: Localizing Program Logical Errors Using Extraction of Knowledge from Invariants. LNCS 6630, pp. 124–135 (2011)
24. Fouladgar M.H., Minaei-Bidgoli B., Parvin H.: On Possibility of Conditional Invariant Detection. LNCS 6881(2), pp. 214–224 (2011)
25. Minaei-Bidgoli B., Parvin H., Alinejad-Rokny H., Alizadeh H., Punch W.F.: Effects of resampling method and adaptation on clustering ensemble efficacy. Online (2011)
26. Parvin H., Minaei-Bidgoli B.: Linkage Learning Based on Local Optima. LNCS 6922(1), pp. 163–172 (2011)
27. Parvin, H., Helmi, H., and Minaei-Bidgoli, B., Alinejad-Rokny, H., Shirgahi H.: Linkage Learning Based on Differences in Local Optimums of Building Blocks with One Optima. International Journal of the Physical Sciences 6(14):3419–3425 (2011)
28. Parvin H., Minaei-Bidgoli M., Alizadeh H.: A New Clustering Algorithm with the Convergence Proof. LNCS 6881(1), pp. 21–31 (2011)
29. Parvin H., Minaei-Bidgoli B., Alizadeh H., Beigi A.: A Novel Classifier Ensemble Method Based on Class Weightening in Huge Dataset. LNCS 6676 (2), pp. 144–150 (2011)
30. Parvin H., Minaei-Bidgoli B., and Alizadeh H.: Detection of Cancer Patients Using an Innovative Method for Learning at Imbalanced Datasets. LNCS 6954, pp. 376–381 (2011)
31. Parvin H., Minaei-Bidgoli B., Ghaffarian H.: An Innovative Feature Selection Using Fuzzy Entropy. LNCS 6677 (3):576–585 (2011)
32. Parvin H., Minaei-Bidgoli B., Parvin S.: A Metric to Evaluate a Cluster by Eliminating Effect of Complement Cluster. LNCS 7006, pp. 246–254 (2011)
33. Parvin, H., Minaei-Bidgoli, B., Ghatei, S., Alinejad-Rokny, H.: An Innovative Combination of Particle Swarm Optimization, Learning Automaton and Great Deluge Algorithms for Dynamic Environments. International Journal of the Physical Sciences 6(22): 5121–5127 (2011)
34. Parvin H., Minaei-Bidgoli B., Karshenas H., Beigi A.: A New N-gram Feature Extraction-Selection Method for Malicious Code. LNCS 6594(2):98–107 (2011)
35. Qodmanan H.R., Nasiri M., Minaei-Bidgoli B.: Multi objective association rule mining with genetic algorithm without specifying minimum support and minimum confidence. Expert Systems with Applications, 38(1):288–298 (2011)
36. Bi Y., Bell D., Wang H., Guo G., Guan J.: Combining multiple classifiers using dempster's rule text caractrization. Applied Artificial Intelligence: An International Journal, 21(3):211–239 (2007)

37. Tan S.: An effective refinement strategy for KNN text classifier. Expert Systems with Applications, 30(2):290–298 (2005)
38. Liao Y., Vemuri V.R.: Use of K-Nearest Neighbor classifier for intrusion detection. Computers & Security, 21(5):439–448 (2002)
39. Chikh M.A., Saidi M., Settouti N.: Diagnosis of Diabetes Diseases Using an Artificial Immune Recognition System2 (AIRS2) with Fuzzy K-nearest Neighbor. Journal of Medical Systems, Online (2011)
40. Liu D.Y., Chen H.L., Yang B., Lv X.E., Li L.N., Liu J.: Design of an Enhanced Fuzzy k-nearest Neighbor Classifier Based Computer Aided Diagnostic System for Thyroid Disease. Journal of Medical Systems, Online (2011)
41. Arif M., Malagore I.A., Afsar F.A.: Detection and Localization of Myocardial Infarction using K-nearest Neighbor Classifier. Journal of Medical Systems, 36(1):279–289 (2012)
42. Mejdoub M., Amar C.B.: Classification improvement of local feature vectors over the KNN algorithm. Multimedia Tools and Applications, Online (2011)
43. Aronson A.R.: Exploiting a Large Thesaurus for Information Retrieval. RIAO: 197–217 (1994)
44. Scott S., Matwin S.: Text Classification Using WordNet Hypernyms. In: Use of Wordnet in natural language processing systems, pp. 38–44 (1998)
45. Yang, T.: Computational Verb Decision Trees. International Journal of Computational Cognition, pp. 34–46 (2006)
46. Munkres, J.: Algorithms for the Assignment and Transportation Problems. Journal of the Society for Industrial and Applied Mathematics, 5(1):32–38 (1957)

Rule Based Case Transfer in Tamil-Malayalam Machine Translation

S. Lakshmi and Sobha Lalitha Devi

AU-KBC Research Centre, MIT Campus of Anna University, Chennai, India

sobha@au-kbc.org

Abstract. The paper focuses on the rule based case transfer, which is a part of the transfer grammar module developed for bidirectional Tamil to Malayalam Machine Translation system. The present study involves two typologically close and genetically related languages, namely Tamil and Malayalam. We considered the basic construction of sentences which is highly dependent on the case systems. The rules were written by taking into consideration the Postpositions and cases in the languages. A parallel corpora was chosen and a deep analysis of the case transfer patterns were done and rules were written to sort out the case changes that happens when translating from one language to another. We have also considered copula transfer in our approach. Web data was used for evaluation and the results were encouraging.

Keywords: Case suffixes, Dravidian languages, machine translation.

1 Introduction

One of the main components of the machine translation system is the transfer grammar that transfers an intermediate representation of the source language to an intermediate representation of the target language. The transfer grammar constitutes of lexical level transfer and structural transfer. In our approach case transfer is taken into consideration. Cases have been used in theChomskyan framework to trigger movement. In Dravidian languages, grammatical relations and semantic roles are usually explained with the help of case suffixes. Case is most easily observed and studied in languages that have a rich case morphology.

Tamil and Malayalam are closely related to each other in grammar and vocabulary than the other two Dravidian languages, Kannada and Telugu. Malayalam is highly influenced by Sanskrit language at lexical, grammatical and phonemic levels were as Tamil is not. The Noun morphology is same in both the languages as the word may contain the root alone or root with suffixes attached to it. Agglutination is widely seen in Tamil and Malayalam. In Tamil and Malayalam the case markers are seen attached to the noun and pronoun information. Postpositions are also seen attached to it. In traditional analysis, there is always a clear distinction made between postpositional

morphemes and case endings. Both the languages belong to the category of nominative-accusative languages. The Tamil verbs inflect for person, number and gender whereas Malayalam verbs do not take person, number and gender termination. Hence the gender marking of the noun is not a relevant feature when Malayalam language is considered. Tamil nouns inflect for case, number (singular and plural) and gender. So when translating from Tamil to Malayalam the verb PNG marker is subdued. A variety of case changes have been observed in the two languages and rules have been formulated. Consider the following example

An accusative dropping was noted when moving from Tamil to Malayalam.

1. *Ta: avan panthai eduthaan*
he ball-acc take-past+3sm
Ml: avan panth eduthu
he ball-nom take-past
(He took the ball.)

In the above example 1 the accusative marking in Tamil is being mapped to nominative case in Malayalam. Malayalam is a language in which only animate objects are marked with accusative case [9]. Rules have been written to handle the accusative drop.

The syntactic difference between languages can be studied to identify an underlying word order in the source language that might be similar to the target language word order. Many approaches have incorporated syntactic information within statistical machine translation systems to obtain better results. Lavie has presented a Stat-XFER, a general search based and syntactic driven framework for developing MT systems [6]. Carbonell, J. G. et al., [1] have developed knowledge based MT by combining syntactic and semantic information to produce an intermediate knowledge representation of the source text which is then generated in the target language. Dave, S., et al., [2] studied the language divergence between English and Hindi and its implication to machine translation between these languages using the Universal Networking Language (UNL).Koehn et al., [4] showed heuristic learning of phrase translations from word-based alignments and lexical weighting of phrase translations leads to significant improvement in translation accuracy. To handle syntactic differences, Melamed [8] proposes methods based on tree-to-tree mappings.Sobha et al., [16] described syntactic structure transfer in a Tamil-Hindi Machine Translation system using hybrid approach where they learned the structures from clause identified parallel data and incorporated it into a rule based system. Sobha et al., [17] has also used a rule-based approach to transfer nominal constructions from Tamil to Hindi. Case transfers from English to Hindi and vice versa has been approached by Sinha [13,14] and case transfer pattern analysis from Hindi to Tamil MT was done by P. Pralayankar et al.,[10].

The paper is organized as follows. In the next section we give a detailed description of various transfers that happen in the Tamil-Malayalam Machine Translation system such as syntactic structure transfer, case transfer and copula transfer. Then we have briefly explained our approach and the computational aspect. The results for the case transfers and conclusion section follows.

2 Types of transfers

Following transfers can happen in transfer grammar module.

1. Syntactic Structure Transfer,
2. Case Transfer, and
3. Copula Generation.

2.1 Syntactic Structure Transfer

The goal of this syntactic structure transfer is to improve the translation grammatically and to give the naturalness to the target language structures [16]. Tamil and Malayalam has similarity at the basic structure level, hence we have given more importance to the lexical level transfers.

2.2 Case Transfer

Lehmann classifies the Tamil case system into 9 cases [5] and Malayalam has been classified to 7 cases [12]. We have done a mapping of the case systems in the two languages and represented it in the table below.

Table 1. Case mapping.

Case	Tamil	Malayalam
Nominative	NULL	NULL
Accusative	Ai	e
Dative	Kku	kk,n
Instrumental	aal, kontu	aal,kont
Locative	il, itam	il,thth
Ablative	Iliruntu	ilninn
Benefactive	Ukkaaka	kkaayi
Sociative	ootu, utan	ot
Genitive	utaiya, in, atu	nte,ute

To analyse the case transfers we have chosen a parallel corpora. In the sections below a detailed description of case transfers is considered by looking into each specific case.

(a) Nominative Case

The nominative case in Tamil and Malayalam is unmarked. A nominal case is identified by the subject of a sentence in its unmarked form. Nominative noun can function as agent and experiencer as shown in example 2.

2. *Ta: avaL aluthaaL*
she-nom cry-past+3sf

Ml: avaL karanju
she-nom cry-past
(She cried.)

(b) Accusative Case

The accusative marker usually follows the object. The accusative case in Tamil marks the direct object noun phrase of a transitive verb. The accusative marker is 'ai' in Tamil and 'e' in Malayalam.

3. *Ta: meri avanai paarthaaL*
Mary-nom him-acc see-past+3sf
Ml: meri avane kandu
Mary-nom him-acc see-past
(Mary saw him.)

An accusative drop was noted when moving from Tamil to Malayalam. Consider the example given below.

4. *Ta: avan panthai eduthaan*
he-nom ball-acc take-past+3sm
Ml: avan panth eduthu
he-nom ball-nom take-past
(He took the ball.)

In Malayalam the accusative suffix is usually dropped in a sentence where the subject- object distinction is clear [11]. In Tamil when the direct object is human, the accusative marker is obligatory, but when non-human object occurs accusative marker signals definiteness [19]. Mohanan has observed that in Malayalam language only animate objects take accusative markers. In the above examples we can see that in example 3 accusative case in Tamil is mapped to accusative in Malayalam and in example 4 the accusative case in Tamil is being mapped to nominative case in Malayalam.

Consider the example 5 given below.

5. *Ta: avaL ammaavai velai ceyyavethaaL*
she-nom mother-acc job do-past-caus+3sf
Ml: avaL ammaye koNt joli ceyyiccu
she-nom mother-acc psp job do-past-caus
(She made her mother work.)

Here the accusative case in Malayalam is marked by the addition of a postposition (koNt) which represents an agentive role.

(c) Dative Case

The dative suffix 'kku' in Tamil is transferred to 'kk' or 'n' in Malayalam. A case divergence has been noted for dative and genitive markers in Malayalam. It was observed by Asher et al., that in Malayalam language dative 'n' occurs with noun roots

ending in 'an' and dative 'kk' occurs with other singular nouns and all plurals. Consider example 6 given below.

6. *Ta: ainth manikin pujai natakkum*
Five o'clock-dat worship-nom happen-fut
Ml: anchu maNikk puja natakkum
Five o'clock-dat worship-nom happen-fut
(The worship will happen at five'oclock.)

Here the dative case in Tamil is mapped to dative in Malayalam. Given below are some exceptions.

7. *Ta: naalaikku paritcai thotankukirathu*
Tomorrow-dat exam start-fut
Ml: naale pareeksha thutangum
Tomorrow-nom exam start-fut
(Exam will start tomorrow.)

In example 7 dative case gets mapped to the nominative case. In Malayalam the subject acquires a dative case only if there is no nominative noun in the sentence.

8. *Ta: raaman vittirkku centran*
Raman-nom house-dat go-past+3sm
Ml: raaman vittil poyi
Raman-nom house-loc go-past
(Raman went home.)

In example 8 dative case of the object gets mapped to locative case. Here the object denotes destination and hence in Malayalam locative marker is used to give the sense.

9. *Ta: amma kathavukku pinnati nintral*
Mother-nom door-dat behind-adv stand-past+3sf
Ml: amma kathakinte pinnil ninnu
Mother-nom door-gen behind-adv stand-past
(Mother stood behind the door.)

In example 9 in Tamil when a dative noun is followed by a place adverb it is found that in Malayalam the dative case gets mapped to genitive. The place adverbs can indicate the static location or movement.

(d) Locative Case

The locative case in Tamil is marked by 2 markers 'il' and 'itam'.Here 'il' specifies 'the place in which' and 'itam' is used for animate nouns to indicate 'with the person'. In Malayalam the locative marker is il and in some cases specified using the marker 'thth'.

10. *Ta: tiraivar hinthiiyil kettaar*
driver-nom hindi-loc ask-past+3sm
Ml:drivar hindiyil codiccu

driver-nom hindi-loc ask-past
(Driver asked in Hindi.)

There were some specific cases were locative marker in Tamil was mapped to Dative and also Genitive +place adverb which is provided in the examples below.

11. *Ta: kaikeyiyitam varam kotuththan*
Kaikeyi-loc boon give+past+3sm
Ml:kaikeyikku varam koduthu
kaikeyi-dat boon give+past
(Gave boon to kaikeyi.)

In example 11 the locative case marker in Tamil changes to dative in Malayalam. In Malayalam there is no distinction for locative marking in animate or inanimate nouns.

12. *Ta: puvitam vantu vanthathu*
flower-loc bee come-past+3sn
Ml: puvinte atuth vant vannu
Flower-gen near-adv bee come-past
(The bee came near the flower.)

In example 12 locative marker in Tamil gets transferred to genitive +adverb in Malayalam.

(e) Genitive

The genitive marker is used to denote possession, relationship and many such semantic relationships. The genitive case is realized by the markers '-in', '-uTaiya' or '-atu', in Tamil. In Malayalam 'nte' and ute are two genitive markers. Genitive nouns are followed by the noun which it modifies.

Given in example 13 is a genitive transfer from Tamil to Malayalam.

13. *Ta: ithu indyaavin vekamaana rayil*
this india-gen fast-adj railway
Ml: ith indyayute vegamulla rayilaaN
this india-gen fast-adj railway-is
(This is India's fastest railways.)

Consider the example below.

14. *Ta: raaja aranmanaiyin araiyil ninRaar*
king palace-gen room-loc stand-past+3sm
Ml: raajav kottarathile muriyil ninnu
king palace-loc+acc room-loc stand-past
(King stood in a room inside the palace.)

In the example 14 above genitive case in Tamil gets replaced to locative+accusative suffix better known as locative copula which is derived from hidden copular verb 'ulla' [11] in Malayalam.

(f) Instrumental

The instrumental suffix 'aal' in Tamil as well as Malayalam is used to specify the means of the cause. Given below is an example for this case suffix transfer. Hence the transfer was from instrumental 'aal' in Tamil to 'aal' in Malayalam.

15. *Ta: raamu penaavaal ezhuthinaan*
ramu pen-ins write-past+3sm
Ml: raamu penayaal ezhuthi
ramu pen-ins write-past
(Ramu wrote with a pen.)

(g) Sociative

The sociative case in Tamil, realized by the markers '-ootu' or '-utan' expresses association or means by which action is done. In Malayalam single marker for sociative case is 'ootu'.

Consider the examples given below

16. *Ta: raaman puthakathotu vanthaan*
raman book-soc come-past+3sm
Ml: raaman pusthakathote vannu
raman book-soc+acc come+past
(raman came with a book.)

Here the sociative case in Tamil is being mapped to sociative+accusative case in Malayalam.

17. *Ta: kantrutan pacu vanthathu*
calf-soc cow come-past
Ml: kitaavinte_koote pasu vannu
calf-gen psp cow come-past
(The cow came with the calf.)

Here sociative case 'utan' known as bound postposition [5] in Tamil gets changed to genitive case followed by post-position 'koote'.

(h) Ablative Case

The ablative case, marked by '-iliruntu' or '-itamiruntu', is transferred to 'ilninn' in Malayalam. In Tamil 'iliruntu' represent the motion from inanimate object and '-itamiruntu' represents the motion from animate object. It was noted by Asher that in Malayalam there is no such case like ablative but it is locative marker il + postposition ninn which provides the meaning of a 'source'.

An example is given below

18. *Ta: ithu manaaliyilirunthu 50 ki.mI. thoorathil irukkirathu*

this Manali-abl 50 k.m. distance-loc is-present+3sn
Ml: ith manaaliyilninn 50 ki.mI. doorathil aan

this Manali-abl 50 k.m distance-loc is-present
(It is 50 k.m from Manali.)

(i) Benefactive Case

The benefactive case in Tamil, realized by the case marker '-ukkaaka', which is transferred to 'kkaayi' in Malayalam. Given example 19 illustrates the benefactive case transfer.

19. *Ta: enakkaaka tiraivar kaathirunthaar*
me-ben driver wait-past+3sm
Ml: enikkaayi drivar kaathirunnu
me-ben driver wait-past
(Driver waited for me.)

2.3 Copula Generation

A copula can be defined as a verb or a verb like word. They are capable of functioning as the main verb, but are grammatically and semantically different from action verbs. Tamil lacks copula. It is included only to convey the meaning. In Malayalam "aak" (form of being) and "unt" are two copulas used. They have been defined as equative and existential copulas [12]. So copula generation have been considered when moving from Tamil to Malayalam.

20. *Ta: avanukku ammaavai pitikkum*
he-dat mother-acc like
Ml :avan ammaye ishtam aaN
he-dat mother-acc like be-pres
(He likes (his) mother.)

3 Our Approach

3.1 Steps Involved in Case Transfer

In the following section we describe the various steps involved to handle the case transfer from Tamil to Malayalam.

- Identify the noun phrases in Tamil source sentence.
- Identify the verbs in the sentence.
- For each noun phrase in the sentence do the following.

1. If the noun word is having nominative case, then the case transfer is nominative itself.
   - If the noun is marked with accusative case and for a list of verbs, then change accusative case suffix into nominative-accusative
     - For another list of verbs and if the current verb is causative, then add postposition "kont" to the accusative marker.

- else transfer accusative case marker “ai” to “e” .

2. If the noun is having dative case then

– For a list of verbs change dative case into locative.
– For a list of verbs if the dative case is followed by a place adverb then change the dative case+ adverb to genitive case+adverb.
– For another list of verbs change dative case to nominative.
– else

(a) If the noun word is singular and masculine then dative case is transferred to 'n'.
(b) If the noun is plural and masculine then dative case is transferred to 'kk'.
(c) If the noun is singular or plural and feminine then dative case is transferred to 'kk'.

3. If the noun is having sociative case then

– For a list of verbs and if the sociative case marker is 'utan'

(a) If the noun word is singular and masculine then change sociative case to genitive 'nte'+psp 'koote'.
(b) If the noun word is plural and masculine then sociative case is transferred to genitive 'ute'+psp 'koote'.
(c) If the noun word is singular or plural and feminine then sociative case is transferred to genitive 'ute'+psp 'koote'.

– For another list of verbs transfer sociative case to sociative +accusative
– else transfer sociative case marker in Tamil 'otu' to sociative case marker 'ot' in Malayalam.

4. If the noun is marked with genitive case then

– For a list of verbs transfer genitive case to locative +accusative.
– else

(a) If the noun word is singular and masculine then dative case is transferred to 'nte'.
(b) If the noun is plural and masculine then dative case is transferred to 'ute'.
(c) If the noun is singular or plural and feminine then dative case is transferred to 'ute'.

5. If the noun is marked with instrumental case then transfer case marker ‘aal|kontu’ then transfers it into 'aal|kont'.
6. If the noun is having ablative case marker 'iliruntu' then transfer it into 'ilninn'.
7. If the noun is marked by benefactive case marker then transfer it into 'kkaayi'.

4 Computational aspect of the case transfer

To give a computational view of the case transfer an example is illustrated below with traces. We have preprocessed the data with morphological information, Parts-of-Speech tagging, chunking information. The example 21 shows how a dative to locative case transfer happens after applying the hand crafted case transfer rules.

21. *seetha kataikku centraal*
sitha-nom shop-dat go-past-3sf
(Seetha went to shop.)

After performing lexical analysis,
Subject : seetha - sitha-NOM(NP1,sg,f)
Object : kataikku- katai-DAT(NP2,sg,n)
Verb : centraal- go-PAST-3.sg.f
On applying the transfer grammar rules,
Subject: seetha (nom)
Object: katai (il)
Verb: cel (...)
During lexical substitution,
Subject: seetha
Object: kata (il)
Verb: pok (...)
So, finally,
seetha katayil poyi.
sitha-Nom shop-Loc go-past
(Sitha went to shop.)

5 Results and Discussion

We evaluated the system with 1000 sentences collected from web articles. The sentences were run through the Tamil-Malayalam Sampark system and the preprocessing errors were taken care as preprocessing errors such as POS tagging errors adversely affected the machine translation output. The correctness of the case transfer depends on the accuracy of the previous modules. Sampark is a platform for Indian language to Indian Language translation [10]. The sentences were evaluated for case transfer and the results are shown in the table below. The results shown are based on real translation system output.

Table 2. Table 2 : Performance of Case Transfer

Case	No. of Cases	Correctly transferred	Accuracy
Nominative	3485	3485	100%
Accusative	300	280	93.3%
Genitive	403	372	92.3%

Case	No. of Cases	Correctly transferred	Accuracy
Dative	213	198	92.95%
Sociative	28	18	64.28%
Locative	650	634	97.53%
Instrumental	76	75	98.68%
Ablative	99	99	100%
Benefactive	28	28	100%
Total	5282	5189	98.23%

From the results it is clear that the case transfer was properly handled and the transfer of sociative case was not properly handled.

Ta: vacista kiraamathil cutaana wannir nirurrukalil kulitha pinpu nInkal viyaas
vasisth village-loc hot water spring-loc bath after you beas

nathiyotu celkintra poluthu colafka vaaykkaal vanthataiyalaam
river-soc go-past-rp solang tributary can-reach

Ml: vasishta graamathil cutvella aruvikalil kulichathinu shesham ningal byaas
vasisth village-loc hot-water spring-loc bath after you beas

nathyilekk ethunna solanga kaivazhi vannucheraam
river-loc+dat go-past-rp solang tributary can-reach

(After taking a bath in the hot water springs of vasisth village you can reach the solang tributary of the beas river.)

Here "nathiyotu" in Tamil is having a meaning towards the river which is represented in malayalam using the locative+dative cases known as allative marker [7]. This was not transferred as per the rules we formulated. Some new rules is to be added where case transfer is not proper.

6 Conclusion

In this paper, we have given a rule based implementation of the case transfer for Tamil-Malayalam machine transfer. On applying the rules into the transfer grammar component and analysing the translation results an improvement was seen in the overall performance of the system. The case transfer has to be improved with more rules. Also various other lexical transfers can be considered to improve the transfer grammar performance.

References

1. Carbonell, J. G., Cullingford, R. E., and Gershman, A. V.: Steps toward knowledge-based machine translation. In: Pattern Analysis and Machine Intelligence, IEEE Transactions on, (4), 376-392 (1981).

2. Dave, S., Parikh, J., and Bhattacharyya, P.: Interlingua-based English–Hindi Machine Translation and Language Divergence. In : Machine Translation, 16(4), 251-304 (2001).
3. Dorr, B. J.: Machine translation divergences: A formal description and proposed solution. In: Computational Linguistics, 20(4), 597-633 (1994).
4. Koehn, P., Josef O.F, and Marcu, D.: Statistical Phrase-Based Translation. In: HLT/NAACL'03. pp. 127-133 (2003).
5. Lehmann T., A Grammar of Modern Tamil. Pondicherry Institute of Linguistics and Culture, Pondicherry, (1989).
6. Lavie, A. : Stat-XFER: A general search-based syntax-driven framework for machine translation. In: Computational Linguistics and Intelligent Text Processing, Lecture Notes in Computer Science. 362375, Springer, (2008).
7. Nirenburg, Sergei. : Knowledge-based machine translation. Machine Translation 4.1 5-24 (1989).
8. Melamed D., Statistical Machine Translation by Parsing. In: ACL, (2004).
9. Mohanan, K. P.,.: Grammatical relations and anaphora in Malayalam. In: Diss. Massachusetts Institute of Technology, (1981).
10. Pralayankar P., Kavitha V., and Sobha L.: Case Transfer pattern from Hindi to Tamil MT. In: PIMT Journal of Research. vol. 2. No. 1, pp. 26-31 March-August (2009).
11. Ravi Sankar S Nair. : A Grammar of Malayalam, www.languageinindia.com/ravisankarmalayalamgrammar.pdf.
12. R.E.Asher and T.C.Kumari. : MalayalamRoutledge, London and New York, (1996).
13. R.M.K. Sinha and Anil Thakur. : Translation Divergence in English-Hindi MT. In: EAMT, Budapest, Hungary (2005).
14. R.M.K. Sinha and Anil Thakur. : Pre-/post-positions Selection in Text Generation for Hindi and other Indian Languages for Translation from English. In: International Symposium on Machine Translation, NLP and Translation Support System, pp: 40-45 New Delhi, (2004).
15. Sangal R.: Project Proposal to Develop Indian Language to Indian Language Machine Translation System.IIIT Hyderabad, TDIL Group, Dept. Of IT, Govt.of India, (2006).
16. Sobha Lalitha Devi, R. Vijay Sundar Ram, Pravin Pralayankar and T. Bakiyavathi. : Syntactic Structure Transfer in a Tamil to Hindi MT System - A Hybrid Approach. In: A. Gelbukh (Ed), Computational Linguistics and Intelligent Text Processing, Springer LNCS Vol. 6008. pp 438 – 450, (2010a).
17. Sobha, Lalitha Devi., Kavitha V., Pralayankar P., Menaka S., Bakiyavathi T., and Vijay Sundar Ram R., Nominal Transfer from Tamil to Hindi. In: International Conference on Asian Language Processing (IALP), Harbin, China. pp. 270 – 273 (2010b).
18. Sobha Lalitha Devi, Sindhuja G., Vijay Sundar Ram R., Transfer Grammar in Tamil-Hindi MT System. IALP 79-82 (2013).
19. Steever, Sanford: The Dravidian Languages, London: Routledge, (1998).

Assessment Criteria for Benchmarking of Arabic Morphological Analyzers and Generators

Tarek Elghazaly and Abdelmawgoud M. Maabid

Department of Computer and Information Sciences,
Institute of Statistical Studies and Research, Cairo University,
Egypt

tarek.elghazaly@cu.edu.eg

Abstract. Natural language processing applications are based on the morphology part. So they should meet some criteria in order to satisfy the required functionality. Assessing and evaluating of Arabic morphological systems depend on the input words and resulted output according to a predefined criteria to measure and analyze given system in order to study its weakness and strength, trying to find an Arabic morphological analyzer free from all mistakes. In this paper we developed the precise assessment criteria for Arabic morphological analyzers to be applied on a given Arabic morphological analyzers and stemming algorithms by voting, after running them on the sample documents selected as the gold standard.

Keywords: Morphology, Arabic morphology, NLP, morphology assessment criteria, stemmer, morphology benchmarking, analyzer, Arabic analyzer.

1 Introduction

Morphology in linguistics concerns with the study of the structure of words [1]. In other words, morphology is simply a term for that branch of linguistics concerned with the forms words take in their different uses and constructions [2].

Arabic is one of the languages having the characteristics that from one root the derivational and inflectional systems are able to produce a large number of words (lexical forms) each having specific patterns and semantics [3]. The root is a semantic abstraction consisting of two, three, or (less commonly) four consonants from which words are derived through the superimposition of templatic patterns [4]. Unfortunately if understanding is considered, un-diacritized words may make problems of meaning; where many words when they appears in un-diacritized text can have more than one meaning; these different meanings rises problems of ambiguity [5].

In Arabic, like other Semitic languages, word surface forms may include affixes, concatenated to inflected stems. In nouns, prefixes include conjunctions ("و" "and", ف "and, so"), prepositions ("بـ" "by, with", "كـ" "like, such as", "لـ" "for, to") and a determiner, and suffixes include possessive pronouns. Verbal affixes include

conjunction prefixes and negation, and suffixes include object pronouns. Either object or possessive pronouns can be captured by an indicator function for its presence or absence, as well as by the features that indicate their person, number and gender[6]. A large number of surface inflected forms can be generated by the combination of these features, making the morphological generation of these languages a non-trivial task [7].

Natural Languages processing and analysis improved substantially in recent years due to applying data intensive computational techniques [8]. However, state of the art approaches are essentially language specific stemmer (Morphology), considering every surface word in the language [9]. A shortcoming of this word-based analysis of the Arabic language is that it is sensitive to lack of data and information about Arabic words and it morphemes. This is an issue of importance as aligned corpora are an expensive resource, which is not abundantly available for many language analysis levels. This is particularly problematic for morphologically rich languages, where word stems are realized in many different surface forms, which exacerbates the hindering higher level of language analysis.

Morphological analysis can be performed by applying language specific rules. These may include a full-scale morphological analysis, or, when such resources are not available, simple heuristic rules, such as regarding the last few characters of a word as its morphological suffix. In this work, we will adapt some major assessment criteria for measuring advantage or drawback of any Arabic morphological system [10].

2 Background And Previous Work

We believe that this is the first proposed work to sum up assessment criteria for Arabic morphological analyzers and Generators. Several researches talked about building powerful stemmers for the Arabic language with accuracies normally exceeding 90% but none of these stemmers offer the source code and/or the datasets used. It is therefore difficult to verify such claims or make a comparison between different stemmers without having the full description of the proposed method or the source code for the implementation of the algorithm [11]. In this section we review some efforts in this direction.

Mohammed N. Al-Kabi and Qasem A. Al-Radaideh [11] proposed analysis of the accuracy and strength of four stemmers for the Arabic language using one metric for accuracy and four other metrics for strength as following:

- The first metric called empirical evaluation (EE), which represents a percentage of the correct roots produced by the stemmer under consideration.
- The mean number of words per conflation class (MWC) depends on the number of words processed.
- Index compression factor (ICF) represents the extent to which a collection of unique words is reduced (compressed) by stemming.
- Word change factor (WCF) represents the proportion of the words in a sample that have been changed in any way by the stemming process.

– The mean number of characters removed in forming stems (Average CR): Usually strong stemmers remove more characters from words to form stems.

Azze Al-din Al-Mazroui, et al. [12] proposed a specification of morphological analysis system in the Arabic language. In this study the researcher outlined the general characteristic that has to consider during process and building Arabic morphological system in terms of input, analysis and output. The study doesn't provide any criteria or automation to compare different systems.

Dassouki [13] proposed a tabulate items as mechanism for assessing morphological analyzer in terms of development of the system speed, input, output, integrating with other applications and capabilities of analyzing new and non-Arabic words. The study doesn't provide any criteria for these selected terms.

William B. Frakes and Christopher J. Fox [14] evaluated the strength and similarity among, four affix removal stemming algorithms. Strength and similarity were evaluated in different ways, including new metrics based on the Hamming distance measure. Data was collected on stemmer outputs for a list of 49,656 English words derived from the UNIX spelling dictionary and the Moby corpus. The study doesn't provide any criteria for these selected measures and it is specific to English stemmers.

3 Proposed Assessment Criteria of Arabic Morphological Systems

Assessing and evaluating Arabic morphological systems depends on the *input* words and resulted *output* [12] according to a predefined criteria to measure and analyze given system in order to study its weakness and strength, trying to find an Arabic morphological analyzer free from all mistakes. Then we will apply these criteria on some of existing available systems; these criticisms will not detract from its value and effectiveness.

3.1 Input

A very fundamental problem with software testing is that testing under all combinations of inputs and preconditions (initial state) is not feasible, even with a simple product. The input can be considered as bulk of text passed to the system in form of word or phrase fully or partially diacritized.

The possibility of analyzing the modern standard texts

Most western scholars distinguish two standard varieties of the Arabic language: the Classical Arabic (CA) of the Qur'an and early Islamic (7th to 9th centuries) literature, and Modern Standard Arabic (MSA), the standard language in use today [15]. The modern standard language is based on the Classical language. Most Arabs consider the two varieties to be two registers of one language, although the two registers can be described in Arabic as (MSA) and (CA) [16].

The possibility of analyzing the common error words

Common typing errors "common error words" are those words mistyped but are traditionally considered correct; typically a feminine ending character "ـة" written without dots "ـه", the dotless "ى" instead of "ي" and the letter "ا"without hamza instead of "أ"; for e example word "احمد" can be read and understood correctly as "أحمد" while the first one is linguistically mistyped [17].

The possibility of analyzing new words (Neologisms)

Neologisms are often created by combining existing words or by giving words new and unique suffixes or prefixes. Portmanteaux "حقائب السفر" are combined words that are sometimes used commonly. Neologisms also can be created through abbreviation or acronym, by intentionally rhyming with existing words or simply through playing with sounds.

Neologisms can become popular through memetics, by way of mass media, the Internet, and word of mouth, including academic discourse in many fields renowned for their use of distinctive jargon, and often become accepted parts of the language. Other times, however, they disappear from common use just as readily as they appeared. Whether a neologism continues as part of the language depends on many factors, probably the most important of which is acceptance by the public. It is unusual, however, for a word to enter common use if it does not resemble another word or words in an identifiable way.

When a word or phrase is no longer "new", it is no longer a neologism. Neologisms may take decades to become "old", however. Opinions differ on exactly how old a word must be to cease being considered a neologism.

Neologisms analysis in morphological system measures the capability of processing the new Arabic words which can be added later to morphological systems' predefined knowledge base.

Processing of Arabized and transliterated words

Transliteration is a subset of hermeneutics. It is a form of translation, and is the practice of converting a text from one script into another. From an information-theoretical point of view, systematic transliteration is a mapping from one system of writing into another, word by word, or ideally letter by letter. Transliteration attempts to use a one-to-one correspondence and be exact, so that an informed reader should be able to reconstruct the original spelling of unknown transliterated words. Ideally, reverse transliteration is possible.

Transliteration is opposed to transcription, which specifically maps the sounds of one language to the best matching script of another language. Still, most systems of transliteration map the letters of the source script to letters pronounced similarly in the goal script, for some specific pair of source and goal language. If the relations between letters and sounds are similar in both languages, a transliteration may be (almost) the same as a transcription. In practice, there are also some mixed transliteration/transcription systems that transliterate a part of the original script and transcribe the rest [13].

In Arabic transliteration is writing non-Arabic words by Arabic alphabet characters as ‘فاكس’ “Fax” in English and “انترنت” “Internet” In English.

Processing of non-tripartite verbs

Arabic verbs, as the verbs in other Semitic languages, are more complicated than those in most languages. A verb in Arabic is based on a set of three or four consonants called a root (trilateral or quadrilateral according to the number of consonants). The root communicates the basic meaning of the verb, e.g. "كتب" k-t-b "write", "قرأ" q-r-’ "read", and "أكل" ’-k-l "eat". Changes to the vowels in between the consonants, along with prefixes or suffixes, specify grammatical functions such as person, gender, number, tense, mood, and voice.

Arabic words are divided into three types: noun, verb, and particle. Nouns and verbs are derived from a closed set of around 10,000 roots. The roots are commonly three or four letters and are rarely five letters. Arabic nouns and verbs are derived from roots by applying templates to the roots to generate stems and then introducing prefixes and suffixes [6].

Assessing and evaluating Arabic considering the system capability of analyze quadrilateral and quinqueliteral verbs like “طمأن” "Reassure" and all possible cases of their forms of transitivity and weakness [12].

3.2 Output

Morphology output is all possible combination of affixes that produced a valid Arabic word, roots and patterns.

Covering analysis of all input words

- The system should cover all cases of analysis.
- Determine word types (pattern, root, stem and attached affixes) [12].
- Analyzing the words in all domains of the language (Geographic, Historical, Religion, and Math).
- Considering syntactic case of input word (within phrase)

Meet all possible cases for analysis

The system has to assume that the input word is a verb, name and character so it has to determine the followings:

- Verb: has to cover non- tripartite, quadrilateral, quinqueliteral with their forms of transitivity, augmentation, hollow…etc. [4].
- Name: has to cover names, infinitives, adjectives and adverbs.
- Particle: has to cover prepositions, conjunctions, vowel, and vocative particles.

Express grammatical function of the affixes

Affixes are those characters attached to the stem (prefix, suffix and infix) each has its own grammatical alternation of the stem attached.

Ambiguity and overlapping of syntactic cases
Many words in Arabic are homographic [5]: they have the same orthographic form, though the pronunciation is different. There are many recurrent factors that contributed to this problem. Among these factors are:

- Orthographic alternation operations (such as deletion and assimilation) frequently produce inflected forms that can belong to two or more different lemmas.
- Some lemmas are different only in that one of them has a doubled sound which is not explicit in writing. Arabic Form I and Form II are different only in that Form II has the middle sound doubled.
- Many inflectional operations underlie a slight change in pronunciation without any explicit orthographical effect due to lack of short vowels (diacritics).
- Some prefixes and suffixes can be homographic with each other. The prefix t can indicate 3rd person feminine or 2nd person masculine.
- Prefixes and suffixes can accidentally produce a form that is homographic with another full form word. This is termed "coincidental identity"
- Similarly, clitics can accidentally produce a form that is homographic with another full word.
- There are also the usual homographs of uninflected words with/without the same pronunciation, which have different meanings and usually different POS's.

That means determining the lack of morphological knowledge of the word analyst; in case of partially diacritized or non-diacritized words, the ambiguity problem may appear, so, the better is to determine all possible cases of the input word; as an example the work "رب" many be either "رَبٌ" (God) or "رُبَّ" (maybe).

Identifying the root of the word and determining all possible roots for the analyzed word
Right root identification of the input word, and with all generated words the system has to be capable to determine their roots and patterns.

Grammatical errors and misspellings in the context of the expression of results of the analysis
The output representation of the system has to be error free in terms of expression and representation of output result.

Cover all possible cases of syntactic word analyst
The system also should be represent and explain the analysis result of each of analyzed word and there generated words.

Consistency between analyzed word and its patterns
The system should produce correct and consistent patterns for the analyzed and generated words.

The result has to be coming from Arabic dictionary
The system should combine the Arabic morphological rules while processing the word with its knowledgebase to reflect a better analysis and generation which measures the trust of morphological analysis result.

3.3 System Architecture and Design

Percentage of non-reliance on predefined knowledgebase of affixes, roots and patterns
An affix is a morpheme that is attached to a word stem to form a new word. Affixes may be derivational, like English -ness and pre-, or inflectional, like English plural -s and past tense -ed. They are bound morphemes by definition; prefixes and suffixes may be separable affixes. Affixation is, thus, the linguistic process speakers use to form different words by adding morphemes (affixation) at the beginning (prefixiation), the middle (infixation) or the end (suffix) of words.

Percentage of non-reliance on common words (Stop List)
In Natural Language Processing (NLP), stop words are words which are filtered out prior to, or after, processing of natural language data. Any group of words can be chosen as the stop words for a given purpose. Common words (stop word) are the words that are frequently used in Arabic text with the same meaning such as day names, month names, numbers names, adverbs… etc.

Processing Speed
In software engineering, performance testing is in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to measure, investigate, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Performance testing is a subset of performance engineering, an emerging computer science practice which strives to build performance into the implementation, design and architecture of a system.

The processing speed can be measured by how many words processed per second.

Ease of use and integration with larger applications
In engineering, system integration is the bringing together of the component subsystems into one system and ensuring that the subsystems function together as a system. In information technology, systems integration is the process of linking together different computing systems and software applications physically or functionally, to act as a coordinated whole.

- How much the system is capable for use and what are the prerequisites for the system to run.
- The ability to integrate the system within larger applications.

– The ability of modifying some of the system behavior of output or even input procedures and functions. (Customization).
– The ability to add inputs to the system knowledgebase.

Availability and documentation
Software documentation or source code documentation is written text that accompanies computer software. It either explains how it operates or how to use it, or may mean different things to people in different roles.

In terms of Arabic morphological system, it measures the availability of the system and its algorithms for newcomer and researchers considering the cost of commercial systems.

User interface (English-Arabic)
The user interface, in the industrial design field of human–machine interaction, is the space where interaction between humans and machines occurs. The goal of interaction between a human and a machine at the user interface is effective operation and control of the machine, and feedback from the machine which aids the operator in making operational decisions. User interfaces exist for various systems, and provide a means of:

– Input, allowing the users to manipulate a system
– Output, allowing the system to indicate the effects of the users' manipulation

Generally, the goal of human-machine interaction engineering is to produce a user interface which makes it easy, efficient, and enjoyable to operate a machine in the way which produces the desired result. This generally means that the operator needs to provide minimal input to achieve the desired output, and also that the machine minimizes undesired outputs to the human.

There are two major factors for judging morphological system interface as follows:

– The Interface language of system itself.
– The language used to represent the output of the system in case of analysis or generation.

Encoding and word representation
Identifying the character encoding used in the system itself for processing and representing the data. As Arabic letters need to be represented in Unicode set; some systems need to transliterate the input as a preparation for processing step and then revert the transliterated results into Arabic to match user input and user interface.

4 Application of the Proposed Assessment Criteria

Assessments are carried out by executing some of the available Arabic morphological analyzers on a randomly selected Arabic political news article, an Arabic Sport News

article "from Al-Ahram newsletter" and the Chapter number 36 of the Holy Qur'an "سورة يس Surah Yassin" with total of 11000 distinct words. We then manually extracted the roots of the test documents' words to compare results from different analyzers, thus creating our baseline test set. Roots extracted were then checked manually in an Arabic dictionary. Voting weights are assigned to each assessment item (assigned Score) in order to accurately make comparisons between these algorithms. Each assessment item has to be applied and calculated as per the result of applying the analysis to the sample input words. Table 1, shows assessment items where the voting mark of each individual item is assigned score of 100points. Here is the step by step procedure of executing the assessment criteria:

1. Manually extract the roots of the test documents' words.
2. Assign voting mark for each assessment item.
3. Manually check the extracted roots against Arabic dictionary.
4. Apply each assessment item separately on each of Arabic Morphological Analyzer.
5. For the output results, check them manually against Arabic dictionary.

Finally, the assessment factors can be separately applied on each of Arabic Morphological Analyzer where all factors can be assigned score with a maximum value of 100 marks. Each assessment factor will be applied and calculated as per Analyzer result of applying the analysis of the sample document words.

Table 1. Assigned scores of the assessment factors.

Cat.	No.	Assessment Criteria	Score %
Input	1	The possibility of analyzing the standard and modern texts	100
	2	The possibility of analyzing the common error words	100
	3	The possibility of analyzing new words	100
	4	Processing of Arabized and transliterated words	100
	5	Processing of non- tripartite verbs.	100
Output	6	Covering analysis of all input words	100
	7	Meet all possible cases for analysis	100
	8	Express grammatical function of the affixes	100
	9	Ambiguity and Overlapping of syntactic cases	100
	10	Identifying the root of the word and determining all possible roots	100
	11	Grammatical errors and misspellings in the context of the results of the analysis	100
	12	Cover all possible cases of syntactic word analyst	100
	13	Consistency between analyzed word and its patterns	100
	14	The result has to be coming from Arabic dictionary	100
System Architectu	15	Percentage of non-reliance on predefined knowledgebase of affixes	100
	16	Percentage of non-reliance on common words	100
	17	Processing Speed	100

Cat.	No.	Assessment Criteria	Score %
	18	Ease of use and integration with larger applications	100
	19	Availability, documentation and customization	100
	20	User Interface (English - Arabic)	100
	21	Encoding and word representation	100
Sum			2200

5 Experiments and Results

Experiments are done by executing some of existing and available Arabic morphological systems on a randomly selected contemporary Arabic political news article, Arabic Sport News article "from Al-Ahram newsletter" and the first 15 verses of chapter number 36 of the Holy Qur'an "Souraht Yassin". Each test document contains domain specific words and represents contemporary and standard Arabic. The test documents contain 540 distinct token. We manually extracted the roots of the test documents' words to compare results for each stemming algorithm. Roots extracted have been check against Arabic dictionary.

The analysis also show that function words such as "فى" "fi", "من" "min", "بين" "bian" are most frequent words in any Arabic text. In other hand, nonfunctional words with high frequency such as "الإفريقية" "al-afiriqiah", "القمة" "al-Qemah" and other words out of 30 most frequent tokens as shown in table I gives a general idea about the main topic of the article.

Simple tokenization is applied for the text of the gold standard documents can be used to test any algorithm smoothly and correctly.

Table 2. Assessment results.

Factor No.	Morphology System			
	Al-Khalil	***Sarf***	***AMA***	***Khoja***
1	75	NA	80	50
2	85	NA	90	20
3	30	NA	20	0
4	10	NA	5	0
5	90	NA	85	80
6	75	NA	80	70
7	87	NA	85	0
8	92	NA	80	0
9	90	NA	35	30
10	85	NA	95	30
11	85	NA	98	90
12	45	NA	40	0
13	80	NA	95	0
14	86	NA	97	80
15	0	0	0	0

16	0	0	0	0
17	35	0	0	30
18	60	60	30	60
19	70	85	0	70
20	50	50	50	50
21	50	50	10	10
Total	1280	245	1075	670

6 Conclusion and Future Research

The proposed assessment criteria are adapted to measure Arabic Morphological Analyzers with some features intended for integration with lager applications in natural language processing. Many other criteria can be added to the proposed items and may vary in weight and phase of testing; similar to the source code related metrics used for measuring the system as a product.

The stemming algorithms involved in the experiments agreed and generate analysis for simple roots that do not require detailed analysis. So, more detailed analysis and enhancements are recommended as future work.

Most stemming algorithms are designed for information retrieval systems where accuracy of the stemmers is not important issue [18]. On the other hand, accuracy is vital for natural language processing. The accuracy rates show that the best algorithm failed to achieve accuracy rate of more than 65%. This proves that more research is required.

References

1. Kiraz, G.A.: Computational Nonlinear Morphology with Emphasis on Semitic Languages. Studies in Natural Language Processing, ed. I. Branimir Boguraev, T.J. Watson Research Center and L.D.C. Steven Bird, University of Pennsylvania, The Edinburgh Building, Cambridge CB2 2RU, Cambridge, United Kingdom (2004)
2. Beesley, K.R.: Arabic Morphological Analysis on the Internet. In 6th International Conference and Exhibition on Multi-lingual Computing, Cambridge (1998)
3. Buckwalter, T.: Buckwalter Arabic Morphological Analyzer Version 1.0. Linguistic Data Consortium (2002)
4. Watson, J.C.E.: The Phonology and Morphology of Arabic. The phonology of the world's languages, ed. J. Durand, New York, United States: Oxford University Press (2007)
5. Mohammed, A.A.: An Ambiguity-Controlled Morphological Analyzer for Modern Standard Arabic Modelling Finite State Networks, School of Informatics, The University of Manchester (2006)
6. Darwish, K.: Building a Shallow Morphological Analyzer in One Day. In 40th Annual Meeting of the Association for Computational Linguistics (ACL-02), Philadelphia, PA, USA (2002)
7. Soudi, A., V. Cavalli-Sforza, and A. Jamari: A Computational Lexeme-Based Treatment of Arabic Morphology. In Arabic Natural Language Processing Workshop, Conference of the Association for Computational Linguistics (ACL 2001) Toulouse, France (2001)

8. Soudi, A., A.V.D. Bosch, and G.U. Neumann: Arabic Computational Morphology. Knowledge-based and Empirical Methods. Text, Speech and Language Technology, ed. N. Ide et al. Vol. 38, The Netherlands: Springer (2007)
9. Shaalan, K.F. and A.A. Rafea: Lexical Analysis of Inflected Arabic Words using Exhaustive Search of an Augmented Transition Network. Software Practice and Experience, Vol. 23(6) (1993)
10. Roark, B. and R. Sproat: Computational Approaches to Morphology and Syntax, United States: Oxford University Press, New York (2007)
11. Al-Kabi, M.N., Q.A. Al-Radaideh, and K.W. Akkawi: Benchmarking and assessing the performance of Arabic stemmers. Journal of Information Science, Vol. 37(111) (2011)
12. Mazrui, A., et al.: Morphological analysis system specifications. In Meeting of experts in computational morphological analyzers for the Arabic language, Damascus (2010)
13. Desouki, M.S.: Mechanism for assessing morphological analyzer. In Meeting of experts in computational morphological analyzers for the Arabic language, The Arab League Educational, Cultural and Scientific Organisation (ALECSO) - King Abdulaziz City for Science and Technology: Damascus (in Arabic) (2009)
14. Frakes, W.B. and C.J. Fox.: Strength and Similarity of Affix Removal Stemming Algorithms. In Proceedings of the Annual Conference on Research and Development in Information Retrieval, ACM SIGIR Forum (2003)
15. Mushira Eid, C.H.: Perspectives on Arabic Linguistics V: Papers from the Fifth Annual Symposium on Arabic Linguistics. Volume 5: John Benjamins Publishing Company (1993)
16. Elgibali, A., K. Versteegh, and M. Eid: Encyclopedia of Arabic Language and Linguistics. Brill Academic Pub. 3250 (2009)
17. Eid, M., V. Cantarino, and K. Walters: Perspectives on Arabic Linguistics VI: Papers from the Sixth Annual Symposium on Arabic Linguistics, Volume 4, John Benjamins Publishing Company, 238 (1994)
18. Sawalha, M. and E. Atwell.: Comparative Evaluation of Arabic Language Morphological Analysers and Stemmers. In COLING 2008 22nd International Conference on Comptational Linguistics, Manchester (2008)

An Approach for Computing Sentiment Polarity Analysis of Complex Why-type Questions on Product Review Sites

Amit Mishra and Sanjay Kumar Jain

Computer Engineering Department, NIT Kurukshetra, Haryana, India

amitmishrag@gmail.com, skj_nith@yahoo.com

Abstract. Opinion questions expect answers from opinionated data available on social web. Opinion why-questions require answers to include reasons, elaborations, explanations for the users' sentiments expressed in the questions. Sentiment analysis has been recently used in answering why type opinion questions. In this paper, we propose an approach to determine the sentiment polarity of complex why type opinion questions that could be expressed in multiple sentences or could have mixed opinions expressed in them. We apply Rhetorical structure theory to determine discourse structure of why type questions. We use such structure to determine sentiment polarity of why type questions and conduct experiments which obtain better results as compared to baseline average scoring methods.

Keywords: Question answering, information retrieval, natural language processing, natural language understanding and reasoning.

1 Introduction

Question Answering Systems (QASs) provide specific answers to users' questions. Most of the research related to Why-type questions in QASs consults information source based on facts i.e., newspaper, technical documents etc [2, 25, 26]. Such questions ask for some facts or methods e.g., why Roses are red? With the emergence of Web 2.0, there are massive user generated data on the web such as social networking sites, blogs, review sites, etc. [23]. These opinionated data sources contain public opinions which could help the users in making judgment about the products. Hence, they could contain answers to why-type questions such as why should I look for product x? [1, 4, 5, 6]. Such questions are referred as opinion questions [1, 4]. The task of generating answers to these questions requires application of opinion mining techniques along with Natural language processing techniques [1, 2, 4]. Research related to why-opinion questions consider simple why-questions expressed in single sentence [1, 2, 4, 5, 6]. To the best of our knowledge there is no work on complex why-type questions that could be expressed in multiple sentences or could have mixed opinions expressed in them. From literature [1, 2, 4, 5, 6], we find that determining the sentiment polarity of why-questions is a significant phase for generating correct answers as it searches for intention of users

with which he is looking for products. Such analysis would determine type of public comments (positive or negative) required to be presented as answers. Most researchers follow average scoring methods which compute the average scores of words in order to determine the final sentiment scores of objects for the task of opinion mining [3, 4, 5, 11, 12, 13, 24].

Such average scoring methods could fell flat in real life scenario for opinion mining tasks [29, 30, 31, 32, 33]. Average scoring methods could yield inaccurate results when applied on complex why-type questions e.g., "I need mobile with good camera. Why Nokia is a bad choice?" Another example, why movie X is bad even if brad has delivered good performance? The average scoring approach could yield false results in determining sentiment polarity of such questions as "bad" and "good" opinion words will neutralize each other to assign neutral score to questions in terms of sentiment polarity. In such circumstances, there is a natural need to fragment why- question into more important and less important spans in view of opinion mining. In the above example, the overall intention of user is determined through text span "Why movie X is bad" not through "brad has delivered good performance".

Bas Heerschop et al. state that most research done in field of sentimental analysis do not take account of documents important structural feature [25]. The authors use rhetorical structure theory to determine discourse structure of document to perform document level sentiment analysis which gives promising results.

Ziheng Lin et al. state that their discourse parser could be utilized in generating answers to why-questions by recognizing causal relations in text [21]. This motivates us to perform discourse based analysis of why-questions.

We perform discourse based analysis of why-questions through a PDTB-Styled End-to-End Discourse Parser developed by Ziheng Lin et al. [21]. We fragment questions into different text spans i.e. more important and less important spans for opinion mining. We use this relation further to determine sentiment polarity of Why-questions. From literature, we find that SentiWordNet [9], MPQA [7], WordNet [15], and Bing Liu's Opinion Lexicon [19] lexical resources are extensively used in opinion mining.

Most of the words are either absent or having stronger objective scores (neutral scores) in these lexical resources [3]. Such words could behave as opinion words when used in questions. For example, why should I choose the product? Here all the words are strong objective words based on SentiWord Net, MPQA, and Bing Liu's Opinion Lexicon. The existing average scoring methods will classify questions as neutral but it asks for positive opinions about the product. Hence, for the task of sentiment classification, the recompilation of the score is necessary in order to determine correct polarity of why-questions.

We present an approach for finding sentiment polarity of complex opinion why-questions. The complex why type- opinion questions could be expressed in multiple sentences or could have mixed opinions expressed in questions. In summary our contribution is as follows:

1. We fragment why-questions into more important and less important spans using a discourse parser [21] and compute score of why-questions as positive, or negative or neutral on the basis of sentiment scores of words of questions computed using different lexical resources.

2. We propose an algorithm which re-computes sentiment polarity scores of different spans of question and perform better in comparison to baseline average scoring methods [2, 4, 5, and 6] in determining opinion polarity of why-questions.

Rest of the paper is organized as follows Section 2 discuss related work. Section 3 presents our Approach for determining sentiment polarity of Why-questions. We have results and discussion in Section 4. Finally, we have conclusions and scope for future research in Section 5.

2 Related Work

Based on works on opinion question answering [1, 2, 4, 5, 6], we find that question analysis, document analysis, retrieval method and answer processing are the steps in drawing answers to opinion why questions. Output of the question analysis phase has cascade effects on other phases in generating correct answers. Further, we find that question analysis comprises of several sub processes i.e., recognizing entity in question, identifying its aspects, detecting sentiment polarity of question and question form. Determining polarity of why-questions is a significant phase for generating correct answers as it searches for intention of users expressed in questions related to products. Sentiment polarity of opinion questions is determined through identification of opinion bearing words and computation of their polarity score through opinion lexical resources [1, 2, 4, 5, 6]. S Moghaddam et al develop an opinion question answering system in which they consider only adjectives as opinion bearing words for the task of determining sentiment polarity of questions [4, 8]. They use a subset of seed words containing 1,336 adjectives. These words are manually classified into 657 positives and 679 negatives by Hat Zivassiloglov et al. [14]. In another work, Farah Benamara found that adjectives and adverbs work better than adjectives alone for the task of sentiment polarity detection [16]. Muhammad Abuliash et al. use adjectives or adjectives headed by adverbs as opinion bearing words in text documents to produce summary of review documents on the basis of features through semantic and linguistic analysis using SentiWordNet [13]. These researchers ignore nouns and verbs which could also behave as opinion words. Turney found that adjectives, verbs, nouns and adverbs play significant role as opinion bearing words for the task of opinion mining [17]. Jong Hu et al. consult a Japanese polarity dictionary distributed via Alagin forum in their question answering [2].The dictionary is not available in English. Jianxing Yu et al. present an opinion question answering system for products review sites by exploiting hierarchical organization of the product reviews [5]. They use SVM sentiment classifier to determine sentiment polarity of questions. For doing this, they consult the MPQA project sentiment lexicon. Most of the words in MPQA project are objective words such as buy; purchase, choose etc. hence we consider the corpus as not a good choice. SenticNet detect sentiment polarity of single sentence by using machine-learning and knowledge-based techniques [29, 30, 31, 32, 33]. The SenticNet capture the conceptual and affective information in the sentence by using

the bag-of- concepts model. The system assumes that input text is opinionated. It does not deal with multiple sentences.

Hongping Fu et al. classify opinion questions into 8 classes: holder, sentiment, target, reason, comparison, y/n, time and location. With regard to why-questions, opinion why-questions could be divided into two classes: open why- questions (Why-questions with unknown reason) and closed why-questions (Why-questions with reason selection) [1]. Fan Bu classify why questions as questions requiring explanations or opinions of others [22].

3 Proposed Approach

In this section, we determine sentiment polarity of why-questions in order to determine the intention of the users with which they are looking for products. We fragment complex why- questions into more important and less important spans in view of opinion mining and then compute sentiment polarity of why-questions on the basis of polarity of more important text span.

3.1 Segmentation of Why-questions in View for Opinion Mining

The objective of this fragmentation is to fragment questions into different text spans and categorize them into more important and less important text span for opinion mining of questions. We present the algorithm that fragment why- questions into more important and less important spans using a discourse parser [21]. The algorithm is as follows:

1. The question text span is parsed through A PDTB-Styled End-to-End Discourse Parser developed by Ziheng Lin et al. [21]
2. If relation equals Cause, or Conjunction or Contrast choose Arg(2) span as first priority.
3. Else If relation equals Condition or others, then choose Arg(1) span as first priority. The output of this algorithm will be a number of text spans with different priorities.

Example:

"I need a mobile with good sound quality and nice looks. I went to market. I found three good shops. I went to shop number 3. Why should one feel sad finally?"

We see the output file as shown below:

{NonExp_0_Arg1 {NonExp_0_Arg1 I need a mobile with good sound quality and nice looks. NonExp_0_Arg1} NonExp_0_Arg1}{NonExp_1_Arg1 {NonExp_0_Arg2_EntRel {NonExp_1_Arg1 {NonExp_0_Arg2_EntRel I went to market. NonExp_0_Arg2} NonExp_1_Arg1} NonExp_0_Arg2} NonExp_1_Arg1} {NonExp_2_Arg1 {NonExp_1_Arg2_EntRel {NonExp_2_Arg1 {NonExp_1_Arg2_EntRel I found three good shops. NonExp_1_Arg2} NonExp_2_Arg1} NonExp_1_Arg2} NonExp_2_Arg1} {NonExp_3_Arg1 {NonExp_2_Arg2_EntRel {NonExp_3_Arg1 {NonExp_2_Arg2_EntRel I went to

shop number 3. NonExp_2_Arg2} NonExp_3_Arg1} NonExp_2_Arg2} NonExp_3_Arg1}{NonExp_3_Arg2_Cause {NonExp_3_Arg2_Cause Why should one feel sad finally? NonExp_3_Arg2} NonExp_3_Arg2}

For relation Non Exp 3 cause, we see Arg 1 as "I went to shop number 3", and Arg 2 as "Why should one feel sad finally?". Hence we select Arg 2 as more important text span. Hence the overall intention of user with which he is looking for product is expressed in Arg 2 text span "Why should one feel finally?"

3.2 Computation of Sentiment Polarity of why-questions

Polarity of why-questions. We compute sentiment polarity of why-questions through the analysis of more important text span of question and determine the scores on the basis of sentiment scores of opinion words of the text span [1,2,4,6]. From literature surveyed, we find that adjectives, nouns, adverb, verb could behave as opinion bearing words. In this regard, we parse the question text span through the Stanford Parser [10] to determine the part of speech of each word. We remove Pre compiled Stopwords from the question words to get opinion words. We change opinion words to their root form through morphological analysis.

We classify the sentiment polarity (i.e. Positive, or negative or neutral) of Question text span through following steps as discussed below:

Computing score of Opinion word: we compute the score of each opinion word of question text span through methods described in literature using different popular sentiment lexicons ie, SentiWord Net, MPQA, Word Net, Bing Liu Opinion lexicon. We propose a method which performs better in comparison to the discussed methods**.**

Computing score of Question text span (Question Polarity Scoring (QPS)): We take average of scores of words to determine overall sentiment polarity of question text span [5, 7, 8, 10, 12, and 28].

3.2.1 Computing score of Opinion word [10, 12, 28]. In this section, we compute the score of each opinion word of question text span by using different popular opinion lexicons, i.e., SentiWord Net, Bing Liu opinion lexicon, MPQA and Word Net. We recomputed the score of words through our algorithm.

Scoring Method 1 consulting SentiWordNet [11, 12, 20]. SentiWordNet is a dictionary of words where scores (positive, negative or neutral) are assigned in the range 0 to 1 to each synset of WordNet.

We compute score of each opinion word through method discussed in papers [11, 12, 20]. Each tokenized word with determined part of speech in the question text span is allotted a positive or negative score with the help of SentiWordNet. As there could be a number of synsets of the word, the score of word is computed as the average score of all synsets of that word.

The positive score is computed as the average of the positive scores of all the synsets corresponding to that word available in SentiWordNet which have same part of speech as of question text span word. Same is done for calculating negative score. Those words which are not found in SentiWordNetare assigned zero.

WordScore(w) is computed by averaging the score (both positive and negative) of the individual words present in the question text span related to the feature M:

$$WordScore(W) = \frac{1}{n}\sum_{i=1}^{n} posScore(i) + \frac{1}{n}(-\sum_{i=1}^{n} negScore(i))$$

where posScore(i), negScore(i) are the positive, or negative score respectively found as of i-th synset of word in question text span S. n = Total Number ofsynsets of word.

As there are 93.75% of words in SentiWordNetare having stronger objective score [3]. Also most of the words have zero positive and negative score such as choose etc. Hence there is need to recomputed the score of such words.

Scoring Method 2 consulting WordNet [15, 18]. We used the Sentiment Symposium Tutorial: Lexicons**,** prepared by Christopher Potts of Stanford Linguistics for computing the score of a word [18]. WordNet is used here [15]. The WordScore(w) is computed by averaging the score of the individual words (w) present in the question text span related to the feature M:

$$\text{WordScore}(w) = \frac{1}{n}\sum_{i=1}^{n} mws(i)$$

Scoring Method 3 using OpinionFinder [28]. We perform subjectivity analysis of Why questions using OpinionFinder System. Opinion Finder recognizes subjective sentences as well as different aspects of subjectivity within sentences.

Scoring Method 4 consulting Bing Liu Opinion Lexicon. We used Bing Liu Opinion Lexicon prepared by Bing Liu [19]. It provides list of positive words and negative words. It does not contain ambiguous words. Hence the coverage is very low with only 2006 number of Positive words and 4783 number of Negative words. If the number of positive words in Question text span is more than number of negative words, then we classify it as Positive else negative.

Our Method: Our modified Word Scoring methods. In our approach, we search for synonymous words to improve the sentiment polarity of why-questions. From our experiments, we find that MPQA and SentiWordnet is the effective dictionary for the purpose. Our approach is as follows:

1. Calculate score of each argument.
2. We compute the score of opinion word extracted in section 3.2. We calculate the score of the word through following rules. As there are two values for subjective score (strong or weak), and two values of positive score (strong or weak) and two values of negative scores (strong or weak) hence there are (2*2*2=8) combinations. And there is one combination of words not found in corpus. Each word score in each argument is calculated from MPQA dictionary
3. If the polarity of word is positive or negative regardless of its score and strength is strongsubj or weaksubj. Then, final score of word will be made same.
   - Strong positive with strong subj of word has score equivalent to 1.00.
   - Strong positive with weak subj of word has score equivalent to .75.

- Weak positive with strong subj of word has score equivalent to .50.
- Weak positive with weak subj of word has score equivalent to .25.
- The word which is not found in the corpus is assigned score 0.00.
- Weak negative with weak subj of word has score equivalent to -0.25.
- Weak negative with strong subj of word has score equivalent to -0.50.
- Strong negative with weak subj of word has score equivalent to -0.75.
- Strong negative with strong subj of word has score equivalent to -1.00.

4. Else the score of the word is calculated with the help of SentiWord Net.

We update Scoring Method 1 consulting SentiWordNet. We do some extra computation on WordScore(w) if it equals to zero. We compute WordScore(w). If WordScore(w) equals to zero, then we search for other synonymous words falling in same synonymous set. We compute WordScore(w)

For example: if I need average mobile, why should I choose the product X?, Choose is synonymous with take#10, select#1, pick_out#1, prefer#2 opt#1 in sentiWord Net. Hence the updated positive score of the "choose" word is average sum of all positive scores of synonymous words. Same is done for negative score computation.

3.2.2 Computing score of Question text span (Question Polarity Scoring (QPS)). QPS is computed by averaging the score (both positive and negative) of the opinion words present in the question text span related to the feature M:

$$QScore(q) = \frac{1}{n}\sum_{i=1}^{n} WordScore(i)$$

where QScore(q) score of question text span Q which is related to product feature M. WordScore(i) is score found of ith word (w) in question text span S. n = Total Number ofwords in Question text span. Based on value of QScore(q), we determine polarity of question span text q. If QScore(q) is positive, hence question span text q have positive polarity. QScore(q) is negative, hence question span text q have negative polarity. QScore(q) is neutral, hence question span text q is neutral.

We analyze 19 manually constructed opinion why-questions with different structures prepared by our colleagues that could be asked on product review sites [the list of questions are given after reference section]. There is no standard data set for opinion "why" questions to the best of our knowledge. We find accuracy of Question Fragmentation module for opinion mining in Table 1. The details are given after reference section. We do the analysis of the questions and determine their sentiment polarity. We followed evaluation method of authors S. Moghaddam et al. in Table 2 [4]. We do analysis of list of questions and their sentiment polarity detection in Table 3. In Table 3, we present the accuracy observed in different methods.

Table 1. Accuracy of Question Fragmentation module for opinion mining [4].

Method	**Our Method**
Accuracy	60%

Table 2. Analysis of sentiment polarity of 'more important text span' of questions [4].

	Questions	**Method 1**	**Method 2**	**Method 3**	**Method 4**	**Our method**
1.	Why should I buy Nokia?	√	√	X	√	√
2.	Why should I like Nokia?	√	√	√	√	√
3.	Why should I go for Nokia?	√	x	x	x	√
4.	Why should I look for Nokia?	√	x	x	x	√
5.	Why should I accept Nokia?	√	√	√	x	√
6.	Why should I choose Nokia?	x	x	x	x	√
7.	Why should I forget Nokia?	√	x	√	x	√
8.	Why should I get fond of Nokia?	√	x	√	√	√
9.	Why should I overlook Nokia?	√	√	√	√	√
10.	Why should I suggest Nokia?	√	x	√	x	√
11.	Why should I recommend Nokia?	√	x	√	√	√
12.	Why should I propose Nokia?	x	x	x	x	√
13.	Why should I advise for Nokia?	x	x	√	x	√
14.	Why should I need Nokia?	x	x	x	x	x
15.	Why should I feel sad?	√	√	x	x	√
16.	Why should I demand for Nokia?	x	x	x	x	x
17.	Why should I call for Nokia?	√	x	x	x	√
18.	Why should I require Nokia?	√	x	x	x	√
19.	Why should I want Nokia?	x	x	√	x	√
20.	Why should I prefer Nokia?	√	x	√	√	√
21.	Why should I desire for Nokia?	√	x	√	x	√
22.	Why should I opt for Nokia?	x	x	x	x	√
23.	Why should I pick Nokia?	x	x	x	x	x
24.	Why should I select Nokia?	x	x	x	x	√
25.	Why should I wish for Nokia?	x	x	√	x	√
26.	Why should I aspire for Nokia?	√	√	√	√	√
27.	Why Nokia is first choice?	√	x	x	x	√
28.	Why I is inclined towards Nokia?	√	x	√	x	√
29.	Why should I favor Nokia?	√	√	√	√	√
30	Why should I order Nokia?	x	x	x	x	x
31.	Why should I insist for Nokia?	x	x	√	x	√
32.	Why should I neglect Nokia?	√	√	√	√	√
33.	Why should I stop thinking about Nokia?	√	x	x	x	√
34.	Why should I put Nokia out of his mind?	x	x	x	x	x
35.	Why should I feel cheated in the end?	x	x	√	√	√
36.	Why should I be happy?	√	x	√	√	√
37.	Why should I feel satisfied finally?	√	√	√	√	√
38.	Why should one leave Nokia?	√	x	x	x	√
39.	Why should one love Nokia?	√	x	√	√	√

Table 3. Accuracy of different methods [23].

Method	Method 1	Method 2	Method 3	Method 4	Our Method
Accuracy	0.64	0.23	0.53	0.33	0.87

3 Results and Discussions

We analyze the results and get following observations. We find that our proposed Method gives maximum accuracy of 60% in segmentation of Why-questions in view for opinion mining. We re-computes sentiment scores of words to give updated positive and negative scores and determine sentiment polarity of WHY type questions. The computed scores of words through our algorithm exhibit better results with maximum accuracy of 0.87 than the scores assigned to the words in SentiWordNet, MPQA, WordNet, and Bing Liu's Opinion Lexicon in determining sentiment polarity of WHY questions.

1. WSD (word sense disambiguation) - we calculate the average sum of all scores of the word related to a given part of speech in SentiWordNet. Words behave differently in terms of polarity in different context. Hence identification of the word sense and allotting the score of the sense directly could improve the performance of the systems. Such as Why I need camera x? Here, average sum of need word leads to negative polarity.
2. Opinion bearing words- identification of opinion bearing words in the sentence could increase the performance of the proposed system. Our system calculates the scores of all words of the sentences.
3. Discourse analysis – we use PDTB-Styled End-to-End Discourse Parser developed by Ziheng Lin et al. [45] as the accuracy of discourse parser in today's era is not very promising hence it affect our performance.
4. Domain specific lexicon. SentiWord Net, MPQA, Bing Liu lexicon are open domain dictionary. Some domain specific lexicons behave differently in polarity than general domain lexicons. E.g. long. If the camera coverage is long then it is good. But the movie is long it expresses negative sentiments.
5. Informal language. Use of informal language effect the method.
6. Use of knowledge-based techniques for opinion mining- we find from literature [29, 30, 31, 32, 33] that the bag of concept model captures conceptual and affective information and are more suitable for task of opinion mining. We will consider using the same in future and investigate the worthiness of them in determining sentiment polarity of why-questions.

4 Conclusions and Future Works

In this paper, we determine the polarity of the questions that could be single or multiple sentence(s) why-type questions through proposed algorithm. We perform discourse based analysis of why type questions before computing sentiment polarity of question through average scoring method. The segmentation of why-questions and their sentiment determination are dependent on performance of automatic discourse parser. Instead of calculating score for all words, we observe that detecting opinion bearing words and computing their sentiment scores could improve the performance of why-QAS. We know SentiWord Net, MPQA is general domain dictionary hence there should be domain specific learning to use same. We find that requirements of people depend upon their choice, age, time, financial status. Hence capturing their requirements from their browsing history as in recommender systems then presenting good or bad quality of the product or services [36] will be more good option. In future we will use different discourse parsers, patterns, i.e., sentic patterns [34, 35] or textual [37] entailment system, semis-supervised [38] learning to evaluate and compare our methods on different parameters. We will use machine learning methods for the task of sentiment polarity detection of questions as it could be effective in different domains.

References

1. Hongping Fu et al.: Classification of opinion questions. In Proceedings of the 35th ECIR conference, Moscow (2013)
2. Jong-Hoon Oh et al.: Why-question answering using sentiment analysis and word classes. In Proceedings of EMNLP-CoNLL, Korea (2012)
3. Chihli Hung, Hao-Kai Lin: Using Objective Words in SentiWordNet to improve Sentiment Classification for Word of Mouth. IEEE Intelligent Systems 28(2), 47–54 (2013)
4. S. Moghaddam and M. Ester: AQA: Aspect-based Opinion Question Answering. In IEEE-ICDMW, Vancouver, Canada (2011)
5. Yu J, Zha Z-J, Wang M, Chua T-S: Answering opinion questions on products by exploiting hierarchical organization of consumer reviews. In Proceedings of EMNLP conference, Jeju, Korea (2012)
6. L.W. Ku, Y.T. Liang, et al.: Question Analysis and Answer Passage Retrieval for Opinion Question Answering Systems. International Journal of Computational Linguistics & Chinese Language Processing (2007)
7. T. Wilson, J. Wiebe et al.: Recognizing Contextual Polarity in Phrase-level Sentiment Analysis. In HLT/EMNLP (2005)
8. S. Moghaddam and F. Popowich: Opinion polarity identification through adjectives. CoRR, abs/1011.4623 (2010)
9. Esuli and F. Sebastiani: SentiWordNet: A publicly available lexical resource for opinion mining. In Proceedings of LREC-06, the 5th Conference on Language Resources and Evaluation, Geneva, Italy (2006)
10. Stanford Part of Speech Tagger: http://nlp.stanford.edu/software/tagger.shtml
11. Gautam Kumar et al.: Opinion mining and summarization for customer reviews. IJEST Volume 4, Issue 8 (2012)

12. Shaishav Agrawal: Feature based Star Rating of Reviews: A Knowledge-Based Approach for Document Sentiment Classification. International Journal of Hybrid Information Technology, Vol. 5, No. 4 (2012)
13. M. Abulaish, Jahiruddin, M.N. Doja, T. Ahmad: Feature and Opinion Mining from Customer Review Documents. In Proceedings of Pattern Recognition and Machine Intelligence (2009)
14. V. Hatzivassiloglou and K. R. McKeown: Predicting the semantic orientation of adjectives. In Proc. of ACL (1998)
15. C. Fellbaum (ed.): Word Net: An Electronic Lexical Database. MIT Press (1998)
16. Farah Benamara, Carmine Cesarano, Diego Reforgiato: Sentiment Analysis: Adjectives and Adverbs are better than Adjectives Alone. In Proc. of ICWSM "Boulder, CO USA
17. Turney, P.: Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics (2002)
18. Sentiment Symposium Tutorial: Lexicons. http://sentiment.christopherpotts.net/lexicon/ (June, 2013)
19. Bing Liu: A list of positive and negative opinion words or sentiment words for English, http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar, last accessed on 11th June, 2013.
20. K. Denecke: Using SentiWordNetfor Multilingual Sentiment Analysis. In Proceedings of the International Conference on Data Engineering (ICDE 2008), Workshop on Data Engineering for Blogs, Social Media, and Web 2.0, Cancun (2008)
21. Ziheng Lin et al.: A PDTB-Styled End-to-End Discourse Parser. http://wing.comp.nus.edu.sg/~linzihen/parser/
22. Fan Bu: Function-based question classification for general QA. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1119–1128, Massachusetts, USA (2010)
23. S Padmaja et al.: Opinion Mining and Sentiment Analysis - An Assessment of Peoples' Belief: A Survey. International Journal of Adhoc, Sensor & Uboquitos Computing, Volume 4; Issue 1 (2013)
24. Liu, Y., Li, S., Cao, Y., Lin, C.-Y., Han, D., & Yu, Y.: Understanding and summarizing answers in community-based question answering services. In Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008), pp. 497–504, Stroudsburg, PA (2008)
25. B. Heerschop et al.: Polarity Analysis of Texts Using Discourse Structure. In Proc. 20th ACM Intl. Conf. Information and Knowledge Management, ACM, pp. 1061−1070 (2011)
26. R. Higashinaka and H. Isozaki: Corpus-based Question Answering for "why" -Questions. In Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP), pages 198–425 (2008)
27. S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen: What is not in the Bag of Words for "why"-QA? Computational Linguistics (2010)
28. Opinion Finder : http://mpqa.cs.pitt.edu/opinionfinder/
29. S. Poria, A. Gelbukh, A. Hussain, N. Howard, D. Das, S. Bandyopadhyay: Enhanced SenticNet with Affective Labels for Concept-based Opinion Mining. IEEE Intelligent Systems, vol. 28, issue 2 (2013)
30. S. Poria, A. Gelbukh, E. Cambria, P. Yang, A. Hussain, T. Durrani: Merging SenticNet and WordNet-Affect emotion lists for sentiment analysis. In IEEE 11th International Conference on Signal Processing, IEEE ICSP 2012, China, Vol. 2, pp. 1251–1255 (2012)

31. S. Poria, A. Gelbukh, E. Cambria, D. Das, S. Bandyopadhyay: Enriching SenticNet Polarity Scores through Semi-Supervised Fuzzy Clustering. In Workshop on Sentiment Elicitation from Natural Text for Information Retrieval and Extraction, SENTIRE 2012, IEEE 12th International Conference on Data Mining Workshops (ICDMW), Belgium, IEEE CS Press, pp. 709–716 (2012)
32. S. Poria, A. Gelbukh, D. Das, S. Bandyopadhyay: Fuzzy Clustering for Semi-Supervised Learning-Case study: Construction of an Emotion Lexicon. Lecture Notes in Artificial Intelligence, N 7629, pp. 73–86 (2012)
33. Poria, S., Agarwal, B., Gelbukh, A., Hussain, A., Howard, N.: Dependency-Based Semantic Parsing for Concept-Level Text Analysis. In Computational Linguistics and Intelligent Text Processing, pp. 113–127, Springer (2014)
34. S. Poria, E. Cambria, G. Winterstein, and G.-B. Huang. Sentic patterns: Dependency-based rules for concept-level sentiment analysis. Knowledge-Based Systems 69, pp. 45–63 (2014)
35. S. Poria, A. Gelbukh, E. Cambria, A. Hussain, and G.-B. Huang: EmoSenticSpace: A novel framework for affective common-sense reasoning. Knowledge-Based Systems, 69, pp. 108–123 (2014)
36. S. Poria, E. Cambria, L.-W. Ku, C. Gui, A. Gelbukh: A rule-based approach to aspect extraction from product reviews. In: COLING, Dublin (2014)
37. Pakray, P., Neogi, S., Bhaskar, P., Poria, S., Bandyopadhyay, S., & Gelbukh, A.: A Textual Entailment System using Anaphora Resolution. System Report. Text Analysis Conference Recognizing Textual Entailment Track Notebook (2011)
38. Poria, S., Gelbukh, A., Hussain, A., Bandyopadhyay, S., Howard, N.: Music genre classification: A semi-supervised approach. Pattern Recognition, pp. 254–263, Springer (2013)

Ad Exchange Optimization Algorithms on Advertising Networks

Luis Miralles Pechuán[1,2], Claudia Sánchez Gómez[1], and Lourdes Martínez Villaseñor[1]

[1] Facultad de Ingeniería, Universidad Panamericana, DF, Mexico

[2] Departamento de Ingeniería y Tecnología de Computadores, University of Murcia, Spain

{lmiralles,cnsanchez,lmartinez}@up.edu.mx

Abstract. Online advertising has seen great growth over the past few years. Advertisers have gotten better results with campaigns targeted at more specific audiences. Ad networks with few visits are unable to create such campaigns and hence are moving forward towards a new model, consisting of a huge global ad exchange market. In this market millions of advertisers compete for the ad space so that their ad will be shown to users upon visiting a page. In selecting the best candidate from all possibilities algorithms able to process advertiser's requirements in tenths of seconds are needed. To face this problem we have developed algorithms using techniques such as threads, AVL trees with hash, multiple node trees or Hadoop technology. Throughout this article we will show the results gained from each algorithm, a comparative performance analysis and some conclusions. We have also proposed some future lines of work.

Keywords: Ad exchange, online advertising algorithms, parallelism, AVL trees, multi-node trees, Hadoop, fuzzy logic.

1 Introduction

Online advertising offers advertisers great advantages when it comes to orienting campaigns to a particularly specified audience or making real-time edits. This explains why more advertisers are choosing to pay for publicity online [1]. Ad networks allow advertisers to post their ads on editor's pages. Editors are the ones with at least one web page and rent space for banners or other such adverts in return for commission.

As time goes on advertisers have been becoming more demanding with the requirements needed to reach an ever more specific audience. Advertisers segment their audiences using various attributes such as city, time, gender, keywords, device or operating system. This is known as microtargeting [2] and reduces the number of visits which can comply with their requirements, but au contraire advertisers pay a

higher price. Micro-targeting consists of segmenting an audience in line with various attributes inn order to be directed towards a small group with the same interest.

Doing this ensures adverts are only shown to users complying with the advertisers requirements and hence are more likely to buy the product. Small ad networks cannot offer such specific campaigns given the fact they do not receive enough visits, and that only a small part actually complies with advertisers requirements. It must also be mentioned that many visitors are not shown a single advert as they do not comply with the set requirements.

That is why it has become vital for small networks to work together to create one large global Ad Exchange Market. Each network is composed of a group of advertisers and a group of editors. In order to manage the exchange we have to take on some tasks such as invoice delivery, private policy [3] and fraud prevention [4] but without a doubt the most important task and what we are going to focus on here is selecting the best candidate from all possible candidates and doing it in the shortest time possible.

Some studies deal with the various factors that should cover the ads-exchange algorithms to assign a value to the Quality Score of each campaign. This research estimates this parameter depending on the performance obtained when other factors were shown [5]. There are also reports that aim to optimize optical price or the best candidate based on a number of parameters by the use of complex mathematical formulas [6]. We will focus on comparing the computational costs of several algorithms whose objective is to select the best candidate in the best possible time. We assigned random values to the Quality Score as assuming that they have already been calculated.

To solve this problem we have developed various solutions. Firstly, we applied parallelism through threads in the C# programming environment; this allows multiple radiuses to be run simultaneously.

We also added fuzzy logic to show adverts to visits that do not exactly comply with advertiser's requirements.

We then proposed other solutions where a tree structure is created in order to reduce the number of comparisons. To do so we used hash coded AVL trees and multiple node trees. These structures make it possible to create tree branches, and hence improve algorithm efficiency.

The final algorithm was developed using the language Pig Latin, from Apache Hadoop. This platform has the necessary tools to simply and efficiently solve Big Data problems.

For each algorithm we created a table of results and then compared them. Finally, we came to some conclusions regarding what we consider the best solution in terms of parameters and then proposed a series of improvements for the future.

2 Description of the Problem to be Solved

Due to the fact there are millions of advertisers and of which each and every one is simultaneously creating multiple campaigns, selecting an ad to be shown is a rather complex task.

To select the best advertiser we have to take them all into account, and give an answer in less than 0.1 seconds [6], so it is of the utmost importance to really design efficient algorithms.

The problem we are trying to solve requires selecting the most adequate campaign for each visit in the shortest time possible. To do this the requirements of each and every campaign need to be analyzed.

Should there be various advertisers who comply with the said requirements; the one with the highest Ad Rank is selected. The Ad Rank is a parameter aiming at better profits for the network but at the same time showing quality ads. The Ad Rank formula is:

$$\text{Ad Rank} = \text{CPC} \times \text{Quality Score}$$

Each platform uses its own method to calculate the Quality Score value, for example Google has never revealed how they calculate theirs.

The advertiser's parameter format is shown in table 1 below, it will be the same for the visit's parameter only with the Quality Score and CPC values removed.

Table 1. Advertisers selected parameter values.

Hour	Browser	Browser Version	OS	OS Version	Parameter N …	Quality Score	CPC
3	Chrome	20.0.1132.47	Macintosh	Intel 10.5	…	0,634	1,695
14	Chrome	22.0.1229.94	Windows	XP	…	0,982	6,088
15	I. Explorer	8.0	Windows	XP	…	0,796	9,370
1	I. Explorer	7.0	Windows	XP	…	0,730	6,856
7	Chrome	22.0.1201.0	Windows	Vista	…	0,545	1,704

The values of each column represented in table 1 are as follows:

1. Hour: Refers to the time of the day the visit was made.
2. Browser: Refers to the browser the visit came from, most commonly Internet Explorer or Chrome.
3. Browser version: This parameter refers to concrete browser version. Browsers are constantly getting faster and more secure version updates.
4. Operating System: The most common ones are Windows, Mac or Linux.
5. OS Version: Just like browsers, OS's have their version e.g. Windows 7, 8 or Mac OS X Lion.
6. Flash version: Some browsers have flash installed. Some versions include 11.3 r31, 10.0 r32 and 10.2 r153.
7. Has flash? : Indicates whether or not parameter uses flash or not.
8. Screen bitrate: This indicates the number of bits needed to show a pixel, usually 32 bits.
9. Screen resolution: Number of pixels by width and height of the on screen image.
10. Country: We can know the country using the users IP number.

11. City: As well as seeing the country of visit origin we can also see the specific city.
12. Language: This indicates the OS language, for example: en, en-us etc….
13. Network address: This refers to the ISP url the user is visiting from.
14. Network name: This refers to the name of the network being used by the user.
15. Access page: The access page is page visited previous to the visit. Most come from search engines but they can also be directly accessed, or through a link.
16. Visit type: User visit types can be direct or referrals using a search engine or any other page type.
17. CPC: Cost per Click. The maximum value an advertiser is willing to pay for an ad to be shown
18. Quality Score: This indicates the ad quality and is calculated based upon many factors such as the number of click per view.

In table 2 we can see configuration options. The numbers in each column represent the advertisers selected parameters; each number corresponds to the parameters described above.

Table 2. Option parameter configuration

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Option 1		X		X						X						
Option 2		X		X						X	X	X			X	
Option 3	X	X	X	X	X			X	X	X	X	X			X	
Option 4	X	X	X	X	X			X	X	X	X	X	X	X	X	X

Google uses supercomputers to solve complex algorithms in a tenth of a second. For example, when we search for "Brazil" at google.com we are given 3,230,000,000 results thanks to the efficient algorithms run on such supercomputers. To solve the problem we have used an Intel(R) Core (TM) i5-2400 CPU @ 3.10 GHz with 16Gb RAM running Windows 7 Pro Service Pack 1 64 bit.

By using this hardware we are going to solve the problem in different ways. We have developed thread code and AVL trees as well as Multi-Node trees within the Microsoft Visual C# 2010, C# Express environment. We have also tried Pig Latin using technology developed by Yahoo called Hadoop. To run Hadoop we have a virtual machine called Hortonworks Sandbox 2.1 on the OS Red Hat, which running Oracle Virtual Box, 4.3.14 r95030 and the aforementioned hardware.

3 Application of Parallelism and Fuzzy Logic within Ad Exchanges

3.1 Applying Parallelism to Ad Exchanges

In today's world we have the power of multi-core processors allowing huge volumes of information to be analyzed. Parallel computing is a processing method using

various instructions at once, as the name suggests, in parallel. This is based on the principle that large scale problems can be divided into smaller ones and hence be resolved simultaneously.

Threads are used in parallel computing, these are tasks that can be done at the same time as others. Different execution threads share a series of resources such as memory space, files or authentication keys.

Parallelism allows various advertisers to be simultaneously compared rather than each visit having to be compared one at a time.

The program we have developed creates 100.000 threads, and provided there is no time limit, can be executed in parallel. Every 100.000 threads can compare 100 campaigns that have been stored in a file along with the visit, making a total of 1.000.000 campaigns. Upon finishing the program, it writes the best solution in a file. We have repeated this step for 1000 visits.

The algorithm we have developed runs using three variables: The option, the number of seconds and the threshold of the degree of similarity. The variable "Option" indicates the number of parameters the advertiser has chosen such as those shown in table 2. For example, option 1 shows the chosen browser, OS and country. The variable "Seconds" indicates the maximum time to calculate a solution and then the variable "Threshold" represents the minimum similarity a visit must have in relation to the advertiser's requirements in order to be shown the ad.

The program pseudo-code is:

```
Begin_Main_Program
From visit = 1 to 1.000
  from j=1 to 100.000
    Create_thread(j);
  from k=1 to 100.000
    run_thread(k);

  While (thread_finish)
    Wait();
  Save_best_solution();
End_Main_Program

Run_Thread_Function
  Read_advertisers_from file(k);
  Compare_advertisers_with_visits();
  if(solution > global_solution)
    global_solution = solution;
  if(Last_thread())
    Write_solution_file(global_solution);
End_Run_Thread_Function
```

3.2 Application of Fuzzy Logic within Ad Exchanges

One of the biggest problems that come up when advertisers configure many a parameter is that very few visits comply with their requirements and hence an ad receives very few views. To increase ad coverage we can apply fuzzy logic. Using this, ads that are very similar but not entirely alike can still be accepted, and hence viewed.

Fuzzy or Heuristic logic is an extension of traditional logic using concepts similar to those of human thought. While traditional logic uses strict boundaries to determine where certain sets belong, for example, "a person is old if they are older than 70", however should a person be 69, they can still be classed as old.

Fuzzy Logic allows us to better adapt ourselves to the real world, and understand such expressions like "It's not very cold" or "You're very young". When it comes to understanding the quantifiers of expressions like "much", "very" or "a few" we use belonging functions to indicate to what extent the element is part of the set. Similarity Matrixes are also used to establish a degree of similarity amongst various elements in a set.

In order to establish the degree of similarity applied to our problem, we have created a series of matrixes that represent the grade of similarity between visit value and configuration value within a campaign on a scale from 0 to 1.

In table 3 we can see the degree of similarity that exists amongst the main languages of visits received by a webpage, given the webpage is in Spanish most visits are from Spanish speakers. In this table we can see the degree of similarity among Spanish speaking countries is very high. In total we have created 12 tables, one for each parameter, on which we wish to apply fuzzy logic.

Table 3. Similarity matrix for OS language parameter.

Language	Ca	En	En-Gb	En-Us	Es	Es-419
Ca	1	X	X	X	X	X
En	0	1	X	X	X	X
En-Gb	0	0.9	1	X	X	X
En-Us	0	0.8	0.7	1	X	X
Es	0	0	0	0	1	X
Es-419	0	0	0	0	0.9	1

3.3 Results Obtained from Fuzzy Logic and Parallelism

With a threshold value of 1 and an option value of 2, the algorithm took a total of 147,3 seconds to compare just one visit with 1.000.000 million advertiser campaigns. If we take the maximum established time of 0.1 seconds into account, we realize that this algorithm is unusable, but we have to take into account the fact that this algorithm is run on a supercomputer with optimized access to files or allows them to be held in memory, so such an algorithm may be viable.

Due to very high times, we have established a maximum number of seconds from which no more threads will be processed. Logically, the higher the number, the more threads can be run, and hence bring a better a solution. With a lower threshold more visits comply with advertisers requirements, giving better results. The results shown in table 4 show the average Ad Rank value. The higher the value, the better the quality of ads displayed.

Looking at table 5 the number of comparisons increases when using fuzzy logic. This is due to the fact that all similarity matrixes have to be run through. Firstly, we look at the lines comparing them with visit parameters and then we look at the columns comparing them with campaign values. Should the results be 3 and 5, the matrix cell [3, 5] will receive a grade of similarity between the two values.

Table 4. Results of the algorithm for parallelization by time using fuzzy logic.

	Threshold	1 Sec	2 Sec	3 Sec	5 Sec	10 Sec	15 Sec	25 Sec
Option 1	0.7	8,77	8,93	9,00	9,05	9,11	9,16	9,38
	0.8	8,78	8,93	9,00	9,05	9,10	9,16	9,40
	0.9	8,78	8,93	8,98	9,06	9,11	9,19	9,37
	1	8,80	8,94	8,99	9,06	9,11	9,17	9,37
Option 2	0.7	8,45	8,62	8,72	8,80	8,88	9,01	9,17
	0.8	8,28	8,54	8,64	8,72	8,85	8,92	9,09
	0.9	6,28	6,89	7,10	7,35	7,65	7,81	7,98
	1	5,01	5,59	6,03	6,26	6,95	7,11	7,54
Option 3	0.7	8,50	8,70	8,79	8,88	8,91	9,05	9,17
	0.8	7,50	7,92	8,11	8,29	8,49	8,57	8,70
	0.9	4,50	5,18	5,55	5,96	6,48	6,71	7,04
	1	0,08	0,19	0,25	0,38	0,51	0,74	1,00
Option 4	0.7	8,10	8,35	8,49	8,64	8,75	8,83	9,01
	0.8	6,58	7,05	7,34	7,63	7,97	8,09	8,25
	0.9	3,58	4,16	4,64	4,93	5,66	5,97	6,29
	1	0,02	0,06	0,05	0,09	0,17	0,19	0,35

Table 5. Number of comparisons for 1.000.000 campaigns using and not using fuzzy logic.

Option	Comparisons Using Fuzzy	Comparisons not using Fuzzy
1	107.600.000	107.400.000
2	1.212.300.000	117.273.810
3	2.035.700.000	134.354.215
4	2.434.900.000	144.191.923

Both "Using fuzzy" and "Not using fuzzy" comparisons are results of comparing visit parameters with 1.000.000 campaign parameters. More time is spent accessing files.

4 Using AVL Trees to Optimize Ad Exchanges

4.1 Developing Algorithms using AVL Trees

To improve computing costs of such algorithms we have employed AVL trees and hash code. AVL trees take their name from the first letter of the surname of its

inventors Adelson-Velskii and Landis. There are binary search trees that satisfy the condition that they are always balanced, so that for each node, the height of the left branch will never differ by more than one unit of the height of the right branch or vice versa.

A binary search tree is a data structure allowing the organization of attribute information; each tree node must comply with the following characteristics: Lower Nodes to the left of a particular node must contain lower values; lower nodes to the right must contain higher values.

For example, let's say the advertiser has decided to configure the following parameters with the following values: Time=21, Browser= Firefox, Browser Version = 14.0.1, OS = Windows, Country = Spain and City = Pamplona. In such a case the chain value will be: "21Firefox14.0.1WindowsXPSpainPamplona". When the hash function is applied to the chain the value becomes: "2C1ECBEA35C21B712410CE7F7D0BB".

Via a hash our algorithm codes the field values of each one of 1.000.000 advertiser's campaigns and then adds them to the AVL tree as nodes. Each node uses an alphanumeric keychain generated by the hash function representing the parameter combination and an attribute with Ad Rank value. A tree must then be created for each of the options, in our case we have four options and hence have created four trees.

4.2 Results Obtained with AVL Trees

The time needed to process 100.000 visits with 1.000.000 advertisers with this algorithm is 1.66 seconds, meaning that the algorithm runs around 9,2 million (9.206.250 to be precise) times faster than the threads. This is due to the algorithm not needing to access any files as the tree can be loaded from memory, and the number of comparisons per visit for option 2 has reduced to 1.172.738.107 with the "Using fuzzy" thread option at only 51.03 with AVL trees.

Table 6. Results obtained from AVL algorithm for 100.000 visits.

Option	Seconds	Average Ad Rank	Average comparisons
1	1,36	9,89	16,65
2	1,55	8,45	30,42
3	1,84	1,24	49,74
4	1,92	0,44	51,03

The average number of comparisons is calculated as the average 100.000 visits. The results are the best possible, given that each and every one of 1.000.000 advertisers has been compared. However, when using threads we had to limit the number by the amount of time taken and hence, the results were not the best possible achieved.

5 Using Multi-Node trees to improve ad performance

5.1 Developing algorithms using Multi-Node trees

Each first level node has a number of children representing possible values a campaign can achieve. Thus if the first campaign parameter has 29 different values, the first level will have 29 children. Should node number 7 on the first level have a second parameter of 12 values, then the node shall have 12 children and so forth for all parameters. The final tree level will contain the Ad Rank value, and just as with the AVL trees we have had to create four Multi-Node trees, one for each configuration option.

To solve the algorithm we tried three different solutions:

1. Unordered trees: These unordered trees consume the least as they do not as they do not have any operations to order. The trees are formed from selected parameters in advertiser's campaigns. The possible values are added to the tree as campaigns are processed
2. Ordered Trees: With ordered trees we applied the same process as with the unordered trees, though we then ordered them alphabetically by parameter name. This was done so that the descending binary search can be used right from the tree root to the leaves to obtain the Ad Rank value.
3. Ordered trees by frequency: In this case, we do the same as the first however we order the trees using the frequency with which an advertiser demands a parameter. If most advertisers configure the time as 13:00 then the most left hand side thread will have this value, and a comparison will be made using this node.

5.2 Results Obtained by Multi-Node Trees

Table 7. Results from Multi-Node trees from 100.000 visits.

		Not ordered		Ordered and using binary search		Ordered by frequency	
Option	Result	Seconds	Average comparisons	Seconds	Average comparisons	Seconds	Average comparisons
1	9,89	0,58	20,59	0,78	37,95	0,53	18,52
2	8,45	1,18	50,90	1,17	75,68	1,03	43,59
3	1,24	1,81	73,25	1,86	99,75	1,69	64,93
4	0,44	1,61	74,98	1,65	102,42	1,65	66,64

As we can see by the number of comparisons the best option is ordering by frequency, the second best are the unordered trees and the third best are the ones ordered by parameter name and then having a binary search applied.

Many tree nodes can be formed by four or five nodes so doing a binary search doesn't make much sense, we can also discard the order by frequency option as the

algorithm has 0.07% less comparisons and hence ordering them into a tree every time a campaign is added would be unjustified.

6 Hadoop Optimizing Ad Exchanges through Apache Hadoop

One of the simplest ways to solve the problem and we can safely assume one of the less puzzling, uses Hadoop, which was famously developed by a Yahoo employee. Apache Hadoop is a framework oriented towards finding solutions to Big Data problems, such is the case in point and the fact it also solves our problem in just a few lines.

This language is oriented to take advantage of the clusters and supercomputers of large companies such as Yahoo, Amazon and Google. These companies use this kind of structure because they run algorithms processing huge amounts of data.

This platform uses two programming languages, Hive and Pig Latin. To solve our problem we used Pig Latin, although it is not as efficient as the trees as it has an additional computing cost 151.2 times higher than AVL trees and 205.9 times higher than Multi-Node trees ordered by frequency, however it allows the problem to be solved in just 10 lines. The code is explained in the program comments below:

```
-- ADVERTISERS
-- Load advertisers table from memory
Anun0 = Load 'default.anunciantes2' USING
org.apache.hcatalog.pig.HCatLoader();
-- For each line we select the columns that interest us
Anun1 = Foreach Anun0 Generate $2, $4, $8, $9, $10, $11, $12, $15,
$19*$20;
-- Then group the lines together to later select the max Ad Rank
Anun2 = Group Anun1 by ($0,$1,$2,$3,$4,$5,$6,$7);
-- Remove groups
Anun4 = For each Anun3 Generate FLATTEN($0),$1;

-- VISITS
-- Load advertisers table from memory
Visitas0 = Load 'default.visitas' USING
org.apache.hcatalog.pig.HCatLoader();
-- For each cell we select the columns that interest us
Visitas1 = For each Visit0 Generate $2, $4, $8, $9, $10, $11, $12, $15;

-- JOINING VISITS AND ADVERTISERS
-- We create a table to coincide with both visit and advertise fields
Visitas2 = Join Visitas1 by ($0,$1,$2,$3,$4,$5,$6,$7), Anun4 by
($0,$1,$2,$3,$4,$5,$6,$7);
-- We then select the columns from those tables that interest us
Res = foreach Visitas2 generate $0,$1,$2,$3,$4,$5,$6,$7,$16;

-- Save answer
store Res into 'Respuestas';
```

In table 8 we can see the results obtained as well as the times needed to obtain them. Tests were done with 100.000 visits and 1.000.000 ad campaigns, and the results obtained are the same as the ones from the AVL and Multi-Node trees. Time is expressed in minutes and seconds, rather than solely seconds as for AVL and Multi-Node trees.

Table 8. Results obtained from the algorithm using Hadoop's Pig Latin

Option	Seconds	Results
1	214	9,89
2	226	8,6
3	260	1,24
4	309	0,44

7 Conclusions

According to the results, the thread option cannot be considered appropriate due to the enormous amount of time required to run the algorithm. One of the reasons behind the elevated time scale is the fact the program takes a long time accessing the 10.000 files used to save advertiser's campaigns. The number of total comparisons is the number for each thread multiplied by the number of threads, coming to a total of 24.349.000.000 comparisons, while using trees it does not exceed 103.

Via Hash, AVL trees give the best results, although they do have two disadvantages, firstly the tree needs to be modified for every single campaign, taking up a lot of time; secondly, these trees are inadequate for Fuzzy Logic use given that upon applying the hash function it gets difficult to compare attributes and establish a degree of similarity as they are coded. In order to implement this kind of logic all possible parameter relations would have to be hash coded with all possible combinations, bringing us to the conclusion that this is an unviable option as it will exponentially increase the number of tree nodes.

Another disadvantage affecting both AVL trees and Multi-Nodes is the computational cost of creating the tree, though this is not really anything to worry about as it can be done offline. That is to say it is not created at the time of a user visit, and hence is not a critical computational cost.

Multi-Node trees have the advantage over AVLs that they can use Fuzzy Logic. This can be done using a simple backtracking algorithm that traverses the tree and changes route should the similarity threshold be overcome. Taking these results into account, it seems that ordering is not a great advantage as looking at the results obtained there is no big difference between the number of neither comparisons nor time consumed.

Finally, if we take the capacity of some of the supercomputers used by technology companies into account, Pig Latin is the best option as its algorithm development code can be summarized in just ten lines. This has the advantage that errors are highly unlikely as well as Fuzzy logic being able to be implemented easily via UDF (User Defined Functions), which are user language implementation methods for both Python and Java programming languages.

8 Future Work

One possible improvement to this algorithm could be adding fuzzy logic; to do this we must make a translation table. If we take into consideration the fact that the three browsers in the table are similar we can then code them with the same code.

To improve comparison block searches upon fuzzy logic application a value could be assigned to each parameter. E.G., I. Explorer 8.0 shall be 7 and I. Explorer 8.0 shall be 9, meaning the similarity between the two will be [7, 9] of the browser matrix. With this a number of comparisons per search will be saved, this can be calculated using the formula: Comparisons: (Lines/2 + Columns/2), assuming the probability of coincidence for all values is the same.

Another improvement could be to keep the ordered by frequency algorithm and instead of ordering it per new campaign, an ordering algorithm shall be applied once per 1,000 new campaigns.

With such formula we can create a tree and then compare each visit with the advertisers formed tree. This in turn would make the program run much faster even though it would also require more lines of code.

References

1. IAB internet advertising revenue report. Obtained from http://www.iab.net/about_the_iab/recent_press_releases/press_release_archive/press_release/pr-122313 (2012)
2. Moe, W. W.: Targeting Display Advertising. London, UK: Advanced Database Marketing: Innovative Methodologies & Applications for Managing Customer Relationships (2013)
3. Neslin (Eds.): Advanced Database Marketing: Innovative Methodologies & Applications for Managing Customer Relationships, Gower Publishing, London (United Kingdom) (2013).
4. G. Johnson: The Impact of Privacy Policy on the Auction Market for Online Display Advertising. Simon School Working Paper No. FR 13-26 (2013)
5. B. Stone-Gross, R. Stevens, A. Zarras, R. Kemmerer, C. Kruegel, and G. Vigna.: Understanding Fraudulent Activities in Online Ad Exchanges. In ACM SIGCOMM Conference on Internet Measurement (IMC) (2011).
6. Y. Chen, P. Berkhin, B. Anderson, and N. R. Devanur: Real-time bidding algorithms for performance-based display ad allocation. KDD (2011)
7. W. Zhang, S. Yuan, and J. Wang: Optimal Real-Time Bidding for Display Advertising. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (2014)

Error Patterns for Automatic Error Detection in Computer Assisted Pronunciation Training Systems

Olga Kolesnikova

Superior School of Computer Sciences, Instituto Politécnico Nacional, Mexico City, Mexico

kolesolga@gmail.com

Abstract. This paper presents error patterns built on the basis of our comparative analysis of American English and Mexican Spanish phonemes and allophones which can be applied in designing the error detection module of a Computer Assisted Pronunciation Training (CAPT) System for teaching American English pronunciation to Mexican Spanish speakers. Error identification is important for an adequate choice of correcting techniques which improves English pronunciation acquisition and helps learners to develop less accented speech. Since automatic individual error detection remains a highly complex computational task, error patterns can enhance the system performance and improve its precision. To the best of our knowledge, error patterns in American English speech generated by Mexican Spanish speakers has not been defined in previous work which was done mainly for Castilian-originated standard Spanish.

Keywords: Error patterns, pronunciation, Mexican Spanish.

1 Introduction

In second language (L2) learning, it is very important to acquire reasonably correct pronunciation. We consider reasonably correct pronunciation because, speaking in terms of general public, it is very hard to develop perfect, native-like L2 pronunciation. Usually, some accent is acceptable whenever the speech of an L2 learner is comprehensible to L2 native speakers.

Correct pronunciation is important not only for L2 learners to be understood adequately, but also for them to understand L2 native speakers. It is a typical problem in L2 learning process that a learner can speak and read, but it becomes a real pain in the neck when it comes to listening comprehension of real-life everyday speech which is usually characterized by high speech, sound reduction, and phonetic variation. Here, the acquisition of correct pronunciation can help since the articulatory and auditory systems are interconnected. A learner is hardly able to recognize a sound which she has never produced due to its absence in her first language (L1). So if a learner has acquired the correct articulation of an L2 sound in its isolate position and in combinations, and has devoted sufficient time to practicing its production, she will be able to recognize it in fluent L2 speech.

In the beginning, we mentioned that it is generally acceptable if an L2 learner develops a reasonably correct pronunciation. However, in some cases, for some activities and occupations, less or even non-accented speech is a requirement. An example of such jobs is operators in call centers. Here, an L2 learner will need more pronunciation training than general language teaching courses can provide, and would look for a specialized course, classes, or software which presents this phonological and phonetic aspect of L2 in more detail.

Nowadays, Computer Assisted Language Learning (CALL) is recognized as a beneficial tool for both L2 teachers and learners. Accessibility in practically all everyday situations, flexibility, adaptability and personalization make CALL systems an excellent instrument in any kind of learning: group and individual, formal and informal, stationary and mobile, in and outside classroom [2, 11, 12, 13].

Many CALL applications are designed to facilitate L2 acquisition in all language aspects: pronunciation, words and their usage, grammar, pragmatics. But there are tutor systems created for Computer Assisted Pronunciation Training (CAPT). A variety of CAPT commercial software can be found online: *NativeAccent*™ by Carnegie Mellon University's Language Technologies Institute, www.carnegiespeech.com; *Tell Me More*® Premium by Auralog, www.tellmemore.com; *EyeSpeak* by Visual Pronunciation Software Ltd. at www.eyespeakenglish.com, *Pronunciation Software* by Executive Language Training, www.eltlearn.com, among others.

Responding to the user's particular need of reducing L2 accent necessary to resolve naturalization and employment issues in English-speaking countries, specialized accent improvement systems have recently been produced. Some examples are *Accent Improvement Software* at www.englishtalkshop.com, *Voice and Accent* by Let's Talk Institute Pvt Ltd. at www.letstalkpodcast.com, *Master the American Accent* by Language Success Press at www.loseaccent.com.

In this paper, we will look at a particular aspect of CAPT systems, namely, their capability of recognizing, or localizing, errors in the learner's speech implemented in the error detection module of the system. Error identification is important for generating an appropriate feedback and corrective exercises to the learner by means of the tutor module of the same system with the purpose of improving the learner's pronunciation and listening comprehension.

Since automatic individual error detection remains a highly complex computational task, error patterns, or error rules, can enhance the system performance and improve its precision. In this work, we define error patterns typical for American English (AE) speech generated by Mexican Spanish (MS) native speakers. The defined error patterns are based on our comparative analysis of AE and MS phonemes and allophones. To the best of our knowledge, such error patterns have not been defined in previous work which was done mainly for Castilian-originated standard Spanish.

The rest of the paper is organized as follows. Section 2 considers the impact of error identification and adequate treatment in the process of L2 acquisition. Section 3 presents the basic architecture of a Computer Assisted Pronunciation Training system and its modules. Section 4 surveys the implementation of Automatic Speech Recognition (ASR) techniques in CAPT systems and briefly describes four essential ASR steps. Section 5 considers two approaches for detecting errors in CAPT systems: general pronunciation assessment and individual error detection. In Section 6 we

present error patterns determined on the basis of our comparative analysis of AE and MS phonemes and allophones, and Section 7 outlines conclusions and future work.

2 Errors in the Process of Second Language Acquisition

Traditional language courses teach pronunciation and auditory recognition of second language phonemes using four basic steps listed as follows, each step is termed twice: first, using general pedagogic terminology, and second, referring to the processes in an intelligent tutor system designed to implement these steps.

At Step 1, which may be called explanation (input), the teacher describes what position the articulatory organs must take and how they must move in order to produce the target sound or sound combination. At Step 2, imitation (output), the learner listens to words with the target sound and repeats them. At Step 3, adjustment (feedback), the teacher identifies, explains, and corrects errors of the learner with relevant exercises until production of the target sound is appropriate depending on the orientation of the course and the learner's level. At Step 4, recognition (assessment), the learner listens to input and discriminates words with the target sound and words without it.

Special attention is paid to correcting the learner's errors at Step 3. Making first articulatory attempts in L2, learners almost always make errors, especially if the phoneme they are practicing at the moment is not present in their L1. In fact, committing and correcting errors is a common aspect of the language learning process. Therefore, it is important for a human teacher or a computer tutoring system to identify errors in the learners' speech, to explain the causes of such error and to offer adequate corrective exercises.

Speaking about intelligent tutor systems, we should mention that their error detecting capacity remains an open question in computer science. Notwithstanding the impressive technological advance we are witnessing now, CAPT systems still require further improvement. The system's capacity to detect errors in the speech of the learner and to offer a relevant feedback —activities performed at Step 3 of the teaching/learning process— is an issue of ongoing research.

In this paper, we focus on this important challenge and address it by defining error patterns to be implemented in the error detection module of a CAPT system to teach American English (AE) pronunciation to native speakers of Mexican Spanish (MS). On the other hand, the same error patterns can be used in the tutor module of a CAPT system in the manner which prevents possible errors and develops new sound generation and auditory recognition skills on the foundation of similar L1 sounds. We believe that such approach will make the process of pronunciation acquisition conscious at all stages (important mostly for adult learners) and free of stress and awkward feelings caused by the fact that learners face the necessity of generating sounds completely alien to them.

3 CAPT Systems

As it was mentioned in the Introduction, Computer Assisted Language Learning (CALL) in general and Computer Assisted Pronunciation Training (CAPT) in particular expand to a great degree opportunities for learners to study independently in a non-judgmental context at their own pace and their preferred location, to view and/or review any part of the materials, to enjoy a variety of practice and to get an individualized corrective feedback.

In this work we are interested in CAPT applications oriented at teaching English as a second language. Although the advantages of such applications are beyond doubt, there are some issues which have not been efficiently resolved yet [9]. These problems include a lack of pedagogical foundation, an emphasis on practicing pronunciation of individual words outside of their context and not in connected speech, insufficient training of suprasegmental features of pronunciation (stress, tone, word juncture) as well as a poor quality of feedback.

The main reason why computer feedback sometimes fails to provide meaningful and relevant assistance to learners is a high complexity of the task which a system has to solve: it must be able to process the learner's speech, identify the pronounced words/phrases and detect errors in them. The area of computer science which deals with these and similar tasks is called Automatic Speech Recognition (ASR), and automatic error detection is a part of ASR.

Quite a lot of research effort has been devoted to solve speech recognition problems; the interested reader may consult some recent ASR advances in [4, 17]. Also, there have been a number of attempts to apply ASR results in CAPT systems perusing the two-sided objective: phonemic recognition of the learner's speech and overall pronunciation assessment or individual error detection [6, 16]; the results obtained at this step are used by the system to generate corrective instructions to the learner. In spite of a progress in improving the quality of computer produced feedback, it is still not as satisfactory as it is expected to be. This work is another attempt to deal with the issue of automatic error detection in CAPT systems in order to improve the precision of the CAPT system feedback.

Usually, in the architecture of CAPT systems, the function of error detection is represented by a separate module. Now we will describe the overall design of such system, explain the functions of each basic module of the system and in Section 5 devoted to error detection we will discuss state of the art techniques implemented in the error detection module of modern CAPT systems.

The basic architecture of a CAPT system includes four principal modules shown in Figure 1. The modules of the system interact with the human learner through interface.

The tutor module simulates the English teacher; its functions are as follows: determine the level of the user (Mexican Spanish speaking learner of English pronunciation in our work); choose a particular training unit according to the learner's prior history stored in the learner's module as data introduced previously via the learner's personal account in the system; present the sound or group of sounds corresponding to the chosen training unit and explain its articulation using comparison and analogy with similar sounds in Mexican Spanish; perform the training stage supplying the learner with training exercises, determining her errors,

generating necessary feedback, and selecting appropriate corrective drills; evaluate the learner's performance; store the learner's scores and error history in the learner's module.

The learner module models the human learner of English; it contains the learner's data base which holds the following information on the learner's prior history: training units studied; scores obtained; errors detected during the stage of articulation training and the auditory comprehension stage.

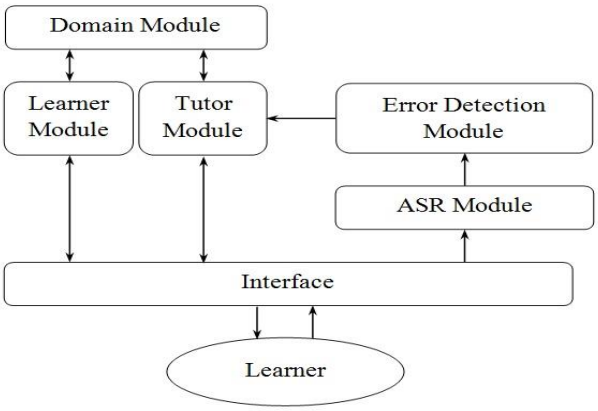

Fig. 1. Basic architecture of a CAPT system

The domain module contains the knowledge base consisting of two main parts: patterns of articulation and pronunciation and auditory perception error patterns characteristic of MS speakers as well as individual error samples; presentation and explanations of sounds, exercises for training articulation and auditory comprehension.

The ASR module is responsible for recognition of the learner's speech.

The error detection module processes the output of the ASR module and identifies pronunciation errors.

4 Automatic Speech Recognition in CAPT Systems

The basic goal of Automatic Speech Recognition (ASR) is to take an acoustic waveform as input and produce a string of words as output. Such analysis involves segmentation of fluent speech into units called phones.

A phone is a speech segment with distinct physical and perceptual (articulatory, acoustic, auditory) features which is a basic unit of phonetic speech analysis, in other words, as we view it, it is a speech sound. The term *phone* is preferred to *sound* in ASR literature although in our opinion both words denote the same entity.

To represent phones, a phonetic alphabet is used. There exist a number of phonetic alphabets, and in this work we use the IPA (International Phonetic Association)

phonetic alphabet; see *Handbook of the International Phonetic Association* [8]. The official website of IPA is at http://www.langsci.ucl.ac.uk.

Another important concept used in ASR is a phoneme which is a "contrastive segment" of speech. The word *contrastive* means that one segment (a phoneme) contrasts with other segments "to make a change in meaning" [3:p.41]. For example, in the four words *cat* [k^hæt], *pat* [p^hæt], *rat* [ræt], *chat* [tʃæt] the only "segment of speech" which differs is the one at the beginning of each word, and this difference produces a change in meaning, so this fact identifies in this case four different phonemes: /k/, /p/, /r/, and /t/.

However, each phoneme can be pronounced in various manners, for example, /k/ in *cat* can be pronounced with aspiration as [k^h] or without aspiration as [k] but such variation does not change the meaning of *cat*, so [k^h] and [k] are not different phonemes, but they are different phones. Phone symbols are written in brackets to distinguish them from phonemes. If two or more phones are realizations of the same phoneme, such phones are called allophones. In our example, [k^h] and [k] are two allophones of the phoneme /k/.

Automatic speech recognition is a complex process consisting of several stages. In summary, a speech signal is first processed to be represented in a computer (this part is called signal processing), and then such representation is analyzed with the purpose of determining to which word or words a given signal corresponds (this part is called signal decoding).

Now, in a more detailed overview, the ASR process can be described by four steps which we are going to discuss now. Each step is considered in a separate subsection. Figure 2 presents a diagram of the four ASR steps.

4.1 Step 1: Speech Waveform Segmentation

At the first stage, a speech signal —an acoustic waveform— is processed to be represented in a computer system. For speech processing, various methods, analog and digital, are used. A common approach is to view a speech signal as a function of time.

However, there are many factors involved in speech production, and speech characteristics change constantly. But if we cut a speech signal into very small intervals of 5 to 25 ms, speech characteristics can be viewed as constants, and intervals of analysis can be mapped to individual phones (at the next stage of ASR).

So at the stage of signal processing, an acoustic waveform is segmented into small pieces called frames. The length of a frame is called the window length. The latter is a parameter, it can be set depending on what information we want to extract from a signal. Also, at this stage which is called the segmentation stage, slices are made in such a way that there is usually a 50% of overlap between two succeeding frames.

4.2 Step 2: Speech Parametrization

At the second stage, each frame is represented by means of a speech vector, or a spectral feature vector. The purpose of this step is to present the speech waveform

under analysis in a compact form and to extract information necessary and sufficient to distinguish one phone (of the inventory, usually about 40 for the English language) from another and filter out acoustic information characteristic of individual speakers.

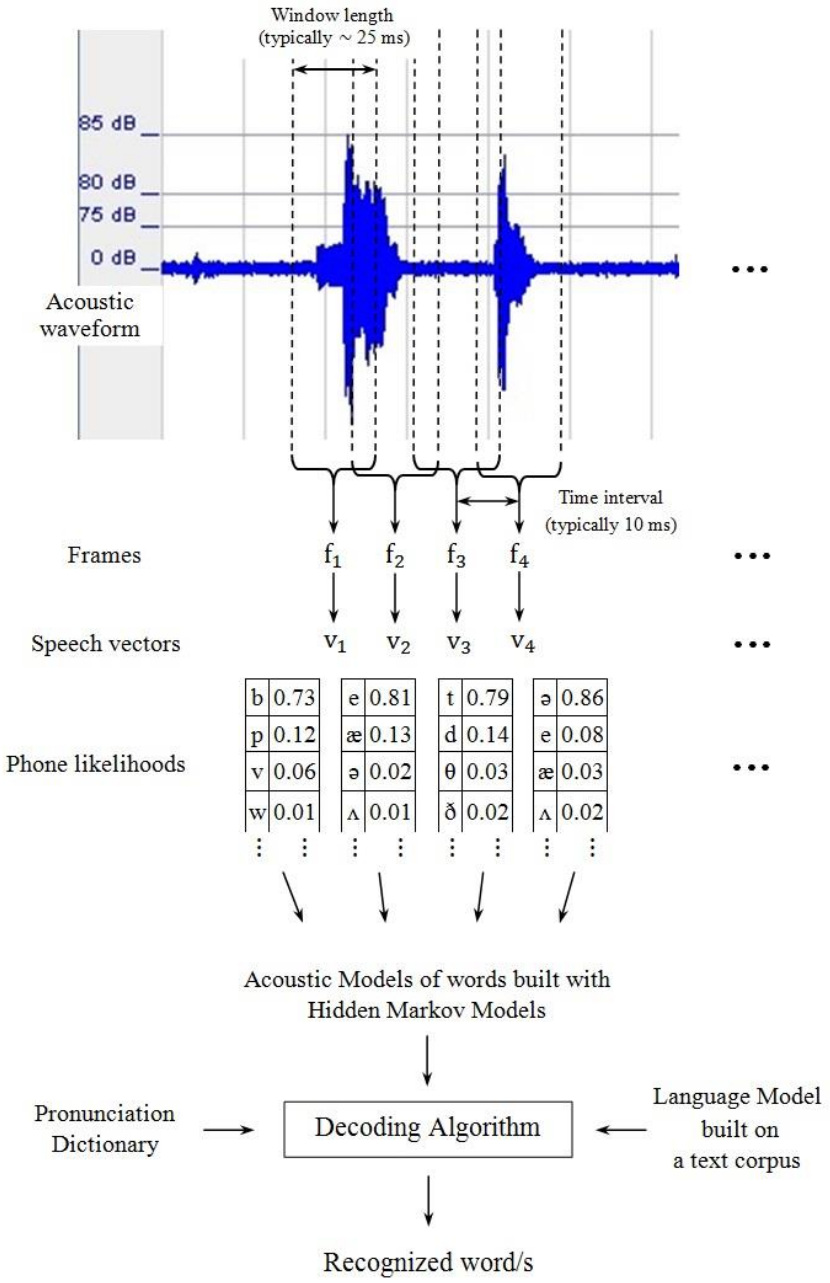

Fig. 2. Automatic speech recognition basic steps

Here we are interested in creating the general acoustic image of a phone "on average", independent of peculiarities of individual pronunciation. This stage of speech processing is called the speech parametrization stage, or the encoding stage.

Various algorithms exist for speech parametrization; the most frequently used of them as well as the simplest one is the procedure based on Mel-frequency cepstral coefficients (MFCCs). The output of this algorithm is a feature vector whose dimensionality is about 40.

4.3 Step 3: Phone Recognition

At the third stage of ASR, speech vectors are mapped to phones or groups of phones; therefore, this stage is called the phone recognition stage. Here, statistical techniques are used such as neural networks or Gaussian models. However, the most common technique used for estimation phone likelihoods is Hidden Markov Models. The underlying idea is to calculate the probabilities for each frame to correspond to each phone in the inventory.

4.4 Step 4: Decoding

The final stage of ASR is called the decoding stage. Now, the probabilities calculated at the previous stage are used to determine to which word in the dictionary a given signal corresponds. At this stage a Viterbi Decoder or a Stack Decoder is used. The input to the decoder is an acoustic model of the utterance to be recognized, a pronunciation dictionary, and a language model. Language model is usually an n-gram model built on a sufficiently large text corpus. The output of the decoder is the recognized word or words.

5 Error Detection in CAPT Systems

In modern CAPT systems, there are two approaches to the learner's pronunciation evaluation and error detection [6]. According to the first approach, the system performs an overall learner's pronunciation evaluation and calculates a measure of **pronunciation assessment**. Within the frame of the second approach, called **individual error detection**, the system detects particular errors of a learner which is a much more difficult issue compared to the first approach due to computational complexity of the ASR task in general and unresolved problems of individual sound recognition in particular, so this issue is still an open question and an area of ongoing research.

5.1 Pronunciation Assessment

Pronunciation assessment is an evaluation of overall impression of the L2 learner's fluent speech. The best known measure is Goodness of Pronunciation (GOP) proposed by Witt [24].

GOP gives a score for each phone of an utterance. To calculate the scores, the GOP algorithm uses the orthographic transcription of the pronounced utterance under analysis and a set of Hidden Markov Models which determine the likelihood $p\left(O^{(q)}|q\right)$ of the acoustic segment $O^{(q)}$ corresponding to each phone q. Then, the quality of pronunciation for any phone p is the duration normalized log of the posterior probability $P\left(p|O^{(p)}\right)$ that the speaker uttered phone p given the corresponding acoustic segment $O^{(p)}$:

$$\mathrm{GOP}(p) = \frac{\left|\log\left(P\left(p|O^{(p)}\right)\right)\right|}{NF(p)} = \frac{\left|\log\left(\frac{p\left(O^{(p)}|p\right)P(p)}{\sum_{q\in Q} p\left(O^{(p)}|q\right)P(q)}\right)\right|}{NF(p)}$$

where Q is the set of all phone models and $NF(p)$ is the number of frames in the acoustic segment $O^{(p)}$. If it is assumed that the likelihood of all phones is the same, i.e., $P(p) = P(q)$, and that the sum $\sum_{q\in Q} p\left(O^{(p)}|q\right)P(q)$ can be approximated by its maximum, the formula becomes

$$\mathrm{GOP}(p) = \frac{\left|\log\left(\frac{p\left(O^{(p)}|p\right)}{\max\limits_{q\in Q} p\left(O^{(p)}|q\right)}\right)\right|}{NF(p)}.$$

5.2 Individual Error Detection

Up to now, attempting to develop a good performance technique for individual error detection, researches have suggested a number of strategies, most representative of which are briefly reviewed in this section.

Weigelt *et al*. [22] used decision trees to discriminate between voiceless fricatives and voiceless plosive using three measures of the waveform. The authors did not apply their results directly to error detection although such application was implied. Later, this method was applied by Truong *et al*. [21] to identify errors in three Dutch sounds /A/, /Y/ and /x/, often pronounced incorrectly by L2-learners of Dutch. The classifiers used acoustic-phonetic features (amplitude, rate of rise, duration) to discriminate correct realizations of these sounds. Truong *et al*. [21] also used classifiers based on Linear Discriminant Analysis (LDA) obtaining good results. Strik *et al*. [20] is another work which experimented with the method in [22] and compared it to other three methods, namely, Goodness of Pronunciation, Linear Discriminant Analysis with acoustic-phonetic features, and Linear Discriminant Analysis with mel-frequency cepstrum coefficients. The analysis was done for the same three Dutch sounds as in [21].

Error detection task was studied for languages other than Dutch. Zhao *et al*. [26] used Support Vector Machines with structural features to identify Chinese pronunciation errors of Japanese learners. Decision tree algorithm was used in the work of Ito *et al*. [10] to identify English pronunciation errors in the speech of Japanese native speakers. The same task was pursued for Korean learners of English in the work of Yoon *et al*. [25] using a combination of confidence scoring at the phone level and landmark-based Support Vector Machines. Menzel *et al*. [14] used the confidence scores provided by a HMM-based speech recognizer to localize English pronunciation error of Italian and German speakers.

It is natural that English as a second language attract attention of researchers who are speakers of other languages. In our work, we are interested in error detection to be implemented in an L2-English pronunciation training system for Mexican Spanish native speakers. For the error detection module of the system we have chosen an approach of error pattern definitions on the basis of comparative analysis of the sound systems of the two languages. Since errors in the patterns have higher probability of occurrence, this fact can improve the overall accuracy of the automatic error detection which by now is not at all satisfactory in state of the art pronunciation training software (see criticism of CAPT systems by Neri *et al*. [16]).

6 Error Patterns for CAPT Systems

Meanwhile the task of individual error detection in any language remains highly complex and not resolved to a satisfactory degree, most state of the art CAPT systems have been designed for pairs of languages, of which one language is usually the native language of a learner, and the other is the second language she is mastering with the help of the CAPT system. In such case, individual error detection is facilitated by the knowledge of typical errors the learner can make.

Typical errors can be encountered theoretically and/or empirically. Theoretical error identification is performed by means of a comparative phonetic analysis of sounds, usually phonemes, and empirical error detection is done based on a study of learner corpora [7].

For example, English learner corpora include recorded interviews, read texts, conversations and other samples of spoken English produced by non-native English speakers. A recognized and oft-used English learner corpus is *Louvain International Database of Spoken English Interlanguage* (LINDSEI, at http://www.uclouvain.be/en-cecl-lindsei.html). This corpus contains oral data produced by advanced learners of English from several mother tongue backgrounds including Bulgarian, Chinese, Dutch, French, German, Greek, Italian, Japanese, Polish, Spanish, and Swedish. It includes almost 800,000 words produced by learners, which represents 554 interviews corresponding to more than 130 hours of recording.

In this paper, we define error patters according to the comparison of American English and Mexican Spanish sounds at the level of allophones which takes into account phonetic processes in both languages. In state of the art works on this theme, analyses are made typically at the level of phonemes only. However, a speaking

person does not produce phonemes (abstract units with the capability of distinguishing meaning), but allophones, i.e., realizations of phonemes in real speech.

Certainly, there exist a very big number of allophones due to language variability depending on phonetic processes, individual articulatory characteristics of a person, his or her educational level, social status, location, age, etc., and it is not feasible to identify all of such allophones and use them in CAPT systems. However, concerning allophones and their acquisition in the process of L2 learning, most frequently met allophones in standard speech should be selected. For American English, General American accent is considered most standard, neutral and free of regional, ethnic or socioeconomic features; it is spoken in many American movies, news, television productions, commercial advertisements, radio programs. Concerning Mexican Spanish pronunciation, the language spoken in university auditoriums, theatre, and mass media is also accepted as the standard accent.

In the next subsection we present the inventory of most frequent allophones of American English (AE) and Mexican Spanish in their standard accents mentioned above, indicating phonemes as well since they are a commonly used tool in L2 pronunciation teaching. Phoneme symbols are given in forward slashes and allophone symbols are put in brackets. Allophones which are pronounced exactly as phonemes are not given, so the sound pronounced as a phoneme can be viewed as the principal allophone of the phoneme. After each phoneme followed by an example word, we give only those most common allophones which acquire additional articulatory and auditory features and thus differ to various degrees from the principal allophone.

The AE and MS phonemes are grouped according to two usual categories, i.e., vowels and consonants; each category is given in its respective subsection. Within each subsection, the phonemes are ordered according to their characteristics, not according to language. This is done with the purpose to show similarities and differenced between AE and MS phonemes and allophones.

We compiled this inventory of phonemes and allophones based on our study of the state of the art works on English and Spanish phonology and phonetics by Whitley [23], Avery and Ehrlich [1], Edwards [5], Quilis [19], Moreno de Alba [15], Pineda, Castellanos, Cuétara, Galescu, Juárez, Llisterri, Pérez and Villaseñor [18].

6.1 Inventory of AE and MS Vowel Phonemes and Allophones

6.1.1. Vowels

- MS high-front /i/ as in *ipo* [ˈipo], nasalized [ĩ] as in *instante* [ĩnˈstaɲ̪te] and *mimo* [ˈmĩmo], palatal semi-consonant [j] as in *pasión* [paˈsjon], palatal semi-vowel [i̯] as in *aire* [ˈai̯re].
- AE high-front tense unrounded /i/ as in *neat* [nit], diphthongized [iɪ] as in *flee* [fliɪ], diphthongized [iə] as in *seal* [siəl], reduced [ə] or [ɪ] as in *revise* [rəˈvaɪ̯z] or [rɪˈvaɪz], lengthened [iː] as in *bee* [biː], semi-lengthened [iˑ] as in *been* [biˑn], shortened [i] as in *beat* [bit].
- AE lower high-front lax unrounded /ɪ/ as in *bit* [bit], reduced [ə] as in *chalice* [ˈtʃæləs], lengthened [ɪː] as in *carrying* [ˈkærɪːŋ].

- MS mid-front /e/ as in *este* [ˈeste], nasalized [ẽ] as in *entre* [ˈẽn̪tre], *nene* [ˈnẽne].
- AE mid-front tense unrounded /e/ as in *ate* [et], diphthongized [eɪ] as in *take* [teɪk], diphthongized and lengthened [eːɪ] as in *say* [seːɪ], diphthongized and semi-lengthened [eˑɪ] as in *name* [neˑim], diphthongized and shortened [eɪ] as in *lake* [leɪk], [i] or [ɪ] as in *Monday* [ˈmʌndɪ].
- AE lower mid-front lax unrounded /ɛ/ as in *get* [gɛt], diphthongized, r-colored and lengthened [ɛːɚ] as in *tear* [tʰ ɛːɚ], diphthongized, r-colored and semi-lengthened [ɛˑɚ] as in *scared* [ˈskɛ̄ˑɚd], diphthongized, r-colored and shortened [ɛɚ] as in *scarce* [skɛ̄ɚs], triphthongized [eɪə] as in *jail* [dʒeɪəl].
- AE low-front lax unrounded /æ/ as in *bat* [bæt], lengthened [æ] as in *bad* [bæːd].
- MS low-central /a/ as in *papa* [ˈpapa], nasalized [ã] as in *ambos* [ˈãmbos].
- AE lower mid-to-back central lax unrounded /ʌ/ as in *above* [əˈbʌv], [ɛ] as in *such* [sɛtʃ], [ɪ] as in *just* [dʒɪst].
- AE neutral mid-central lax unstressed unrounded /ə/ as in *above* [əˈbʌv], lower high-front lax unrounded [ɪ] as in *telephone* [ˈtelɪfon].
- AE mid-central r-colored tense /ɝ/ as in perk [pʰɝk], lengthened [ɝː] as in *sir* [sɝː], semi-lengthened [ɝˑ] as in *learn* [lɝˑn], shortened [ɝ] as in *thirst* [θɝst].
- AE mid-central r-colored lax /ɚ/ as in herder [ˈhɝdɚ], r-dropped [ə] as in *motherly* [ˈmʌðəlɪ].
- AE high-back tense rounded close /u/ as in *boot* [but], diphthongized [uə] as in *stool* [stuəl], diphthongized [uʊ] as in *do it* [ˈduʊɪt], reduced [ʊ] or [ə] as in *to own* [tʊˈon], *to go* [təˈgo], lengthened [uː] as in *blue* [bluː], semi-lengthened [uˑ] as in *food* [fuˑd], shortened [u] as in *loop* [lup].
- AE high-back lax rounded /ʊ/ as in *book* [bʊk], reduced [ʌ] or [ə] as in *would* [wʌd] or [wəd].
- MS mid-back /o/ as in *oso* [ˈoso], nasalized [õ] as in *hombre* [ˈõmbre] or *mono* [ˈmõno].
- AE mid-back tense rounded close /o/ as in *owed* [od], diphthongized [oʊ] as in *go* [goʊ], reduced [ə] as in *window* [ˈwɪndə], diphthongized and lengthened [oːʊ] as in *no* [noːʊ], diphthongized and semi-lengthened [oˑʊ] as in *load* [loˑʊd], diphthongized and shortened [oʊ] as in *coat* [kʰoʊt].
- AE low mid-back lax rounded open /ɔ/ as in *bought* [bɔt], lengthened [ɔː] as in *law* [lɔː], semi-lengthened [ɔˑ] as in *dawn* [dɔˑn], shortened [ɔ] as in *thought* [θɔt], lowered [ɒ] or [ɑ] as in *cot* [kɒt] or [kɑt].
- AE low-back lax unrounded open /ɑ/ as in *pot* [pɑt], rounded [ɒ] as in *got* [gɒt], fronted [a] as in *not* [nat], fronted and rounded [ɔ] as in *father* [ˈfɔðɚ].
- MS high-back /u/ as in *pupa* [ˈpupa], nasalized [ũ] as in *un soto* [ˈũnˈsoto] or *mundo* [ˈmũn̪do], velar semi-consonant [w] as in *cuatro* [ˈkwatro], velar semi-vowel [u̯] as in *auto* [ˈau̯to].
- AE rising low-front to high-front diphthong /aɪ/ as in *kite* [kaɪt], triphthongized [aɪə] as in *I'll* [aɪəl], reduced [ə] *I don't know* [əˈdõʔˈno], lengthened [aːɪ] as in *lie* [laːɪ], semi-lengthened [aˑɪ] as in *find* [faˑɪnd], shortened [aɪ] as in *light* [laɪt], elevated [ɜɪ] as in *ice* [ɜɪs].

- AE rising low-front to high-back diphthong /aʊ/ as in *now* [naʊ], reduced [ʌʊ] as in *house* [hʌʊs].
- AE rising mid-back to high-front diphthong /ɔɪ/ as in *voice* [vɔɪs], lengthened [ɔːɪ] as in *boy* [bɔːɪ], semi-lengthened [ɔˑɪ] as in *noise* [nɔˑɪz], shortened [ɔɪ] as in *exploit* [əksˈplɔɪt].

6.1.2. Consonants

- AE voiceless bilabial stop /p/ as in *pet* [pet], /p/ with aspirated release [pʰ] as in *poke* [pʰoʊk], /p/ with unaspirated release [p̄] as in *spot* [sp̄ɑt], /p/ with nasal release [p̃] as in *stop 'em* [stɑp̃m̩], unreleased [p̚] as in *top* [tɑp̚], lengthened [pː] as in *stop Pete* [ˈstɑpːit], preglottalized [ʔp] as in *conception* [kənˈsɛʔpʃn].
- MS voiceless bilabial unaspirated stop /p/ as in *poco* [ˈpoko].
- AE voiced bilabial stop /b/ as in *bet* [bet], /b/ with nasal release [b̃] as in *rob him* [rɑb̃m̩], unreleased [b̚] as in *rob* [rɑb̚], lengthened [bː] as in *rob Bob* [ˈrɑbːˈbɑbː].
- MS voiced bilabial stop /b/ as in *van* [ban], approximant (spirantized) [β̞] as in *haba* [ˈaβ̞a].
- MS voiced dental stop /d/ as in *dar* [dar], approximant (spirantized) [ð̞] as in *nada* [ˈnað̞a].
- MS voiceless dental unaspirated stop /t/ as in *tío* [ˈtɪo].
- AE voiceless alveolar stop /t/ as in *ten* [ten], /t/ with aspirated release [tʰ] as in *tape* [tʰeɪp], /t/ with unaspirated release [t̄] as in *stop* [st̄ɒp], /t/ with nasal release [t̃] as in *button* [bʌt̃n̩], unreleased [t̚] as in *coat* [kot̚], lengthened [tː] as in *let Tim* [ˈletːˈɪm], dentalized [t̪] as in *eighth* [eɪt̪θ], flapped [ɾ] as in *letter* [ˈleɾə], preglottalized [ʔt] as in *atlas* [ˈæʔtləs], glottal stop [ʔ] as in *button* [bʌʔn], affricated (palatalized) [tʃr̝̊] as in *train* [tʃr̝̊eɪm], affricated (palatalized) [tʃ] as in *eat yet* [ˈitʃət].
- AE voiced alveolar stop /d/ as in *den* [den], /d/ with bilateral release [d◡l] as in *cradle* [kreɪd◡l], /d/ with nasal release [d̃] as in *rod 'n reel* [rɑd̃n̩ril], unreleased [d̚] as in *dad* [dæːd̚], lengthened [dː] as in *sad Dave* [ˈsæːˈdːev], dentalized [d̪] as in *width* [wɪd̪θ], flapped [ɾ] as in *ladder* [ˈlæɾə], affricated (palatalized) [dʒr] as in *drain* [dʒreɪn], affricated (palatalized) [dʒ] as in *did you* [ˈdɪdʒə].
- AE voiceless velar stop /k/ as in *cap* [kæp], /k/ with aspirated release [kʰ] as in *keep* [kʰip], /k/ with unaspirated release [k̄] as in *skope* [sk̄op], /k/ with bilateral release [k◡l] as in *clock* [k◡lɑk], /k/ with nasal release [k̃] as in *beacon* [bik̃n̩], unreleased [k̚] as in *take* [teɪk̚], lengthened [kː] as in *take Kim* [teɪkːɪm], preglottalized [ʔk] as in *technical* [ˈtɛʔknɪk◡l], glottal stop [ʔ] as in *bacon* [beɪʔn̩].
- MS voiced velar unaspirated stop /k/ as in *cama* [ˈkama], palatalized [kʲ] as in *queso* [ˈkʲeso].
- AE voiced velar stop /g/ as in *gap* [gæp], /g/ with bilateral release [g◡l] as in *glee* [g◡li], /g/ with nasal release [g̃] as in *pig and goat* [ˈpɪg̃n̩ˈgot],

unreleased [g̚] as in *flag* [flæg̚], lengthened [gː] as in *big grapes* [ˈbɪˈgːreɪps].

- MS voiced velar stop /g/ as in *gato* [ˈgato], approximant (spirantized) [ɣ̞] as in *el gasto* [elˈɣ̞asto].
- AE voiceless labiodental fricative /f/ as in *fan* [fæn], interdental [θ] as in *trough* [trɑθ], bilabial [ɸ] as in *comfort* [ˈkʌmɸət].
- MS voiceless bilabial fricative /f/ as in *foco* [ˈfoko].
- AE voiced labiodental fricative /v/ as in *van* [væn], devoiced [v̞] as in *have to* [ˈhæv̞tə].
- MS voiceless dental fricative /s̪/ as in *Asia* [ˈas̪ja].
- AE voiceless interdental fricative /θ/ as in *thigh* [θaɪ], voiced [ð] as in *with many* [wɪðˈmenɪ].
- AE voiced interdental fricative /ð/ as in *thy* [ðaɪ], devoiced [ð̞] as in *This is not theirs* [ð̞ɪsɪz ˈnɒʔˈð̞ɛːəz].
- AE voiceless alveolar fricative /s/ as in *sip* [sɪp], palatalized [ʃ] as in *kiss you* [ˈkɪʃju].
- MS voiceless dorosalveolar fricative /s/ as in *sol* [sol], palatalized [ʒ] as in *pues ya* [puˈeʒa], voiced [z] as in *mismo* [ˈmizmo].
- AE voiced alveolar fricative /z/ as in *zip* [zɪp], devoiced [z̬] as in *keys* [kiz̬], palatalized [ʒ] as in *as you* [æˈʒju], stopping [d] as in *business* [ˈbɪdnɪs].
- AE voiceless palatal fricative /ʃ/ as in *mesher* [ˈmeʃə].
- MS voiceless palatal fricative /ʃ/ as in *Xola* [ˈʃola].
- AE voiced palatal fricative /ʒ/ as in *measure* [ˈmeʒə], affricate [dʒ] as in *garage* [gəˈrɑdʒ].
- MS voiced dorsal palatal fricative /ʝ/ as in *yo* [ʝo].
- MS voiceless velar fricative /x/ as in *paja* [ˈpaxa].
- AE voiceless glottal fricative /h/ as in *hat* [hæt], voiced [ɦ] as in *ahead* [əˈɦed], palatalized [ç] as in *hue* [çju], /h/ with glottal release [ʔ] as in *hello* [ʔeˈləʊ], omitted [ø] as in *he has his* [hi hæz ɪz].
- AE voiceless alveo-palatal affricate /tʃ/ as in *chin* [tʃɪn].
- AE voiced alveo-palatal affricate /dʒ/ as in *gin* [dʒɪn].
- MS voiceless palatal affricate /t͡ʃ/ as in *hacha* [at͡ʃa].
- AE voiced labiovelar glide approximant /w/ as in *wed* [wed], aspirated [hw] as in *where* [hweə], devoiced [w̥] as in *twenty* [ˈtw̥entɪ].
- MS voiced alveolar thrill approximant /r/ as in *perro* [ˈpero], devoiced hushing sibilant [r̝̊] as in *ver* [be̞r̝̊], sibilant flap [ɾ] as in *pero* [ˈpeɾo].
- AE voiced alveopalatal liquid approximant /r/ as in *red* [red], devoiced [ɻ̟̊] as in *treat* [tɻ̟̊it], flap [ɾ] as in *very* [ˈveɪ], retroflexed [ɻ] as in *right* [ɻaɪt], back [ɻ̠] as in *grey* [gɻ̠eɪ].
- AE voiced palatal glide approximant /j/ as in *yet* [jet], omitted [ø] as in *duty* [ˈdutɪ], devoiced [j̊] as in *pure* [pʰj̊uə].
- AE voiced alveolar lateral liquid approximant /l/ as in *led* [led], light [l] as in *lease* [lis], dark, velarized [ɫ] as in *call* [kɔɫ], syllabic, also dark [l̩] as in *bottle* [bɑʔl̩], devoiced [l̥] as in *play* [pl̥eɪ], dentalized [l̪̊] as in *health* [hɛl̪̊θ].
- MS voiced alveolar lateral liquid approximant /l/ as in *loco* [ˈloko].

- AE voiced bilabial nasal /m/ as in *met* [met], syllabic [m̩] as in *something* [ˈsʌm̩θɪŋ], lengthened [m:] as in *some more* [sʌˈm:ɔr], labiodentalized [ɱ] as in *comfort* [ˈkʌɱfət].
- MS voiced bilabial nasal /m/ as in *más* [mas].
- MS voiced dental nasal /n̪/ as in *antes* [ˈan̪tes].
- AE voiced alveolar nasal /n/ as in *net* [net], syllabic [n̩] as in *button* [bʌʔn̩], lengthened [n:] as in *ten names* [ten:eɪmz], labildentalized [ɱ] as in *invite* [ɪɱˈvaɪt], dentalized [n̪] as in *on Thursday* [ən̪ˈθɝ͡zde], velarized [ŋ̩] as in *income* [ˈɪŋ̩kəm].
- MS voiced alveolar nasal /n/ as in *nene* [ˈnene], dentalized [n̪] as in *cuanto* [ˈkwan̪to], velarized [ŋ̩] as in *banco* [ˈbaŋ̩ko].
- MS voiced palatal nasal /ɲ/ as in *año* [aɲo].
- AE voiced velar nasal /ŋ/ as in *lung* [lʌŋ], syllabic [ŋ̩] as in *lock and key* [ˈlɒkŋ̩ˈki], alveolarized [n] as in *running* [ˈrʌnɪn], stop [ŋ̩^{k}] or [ŋ^{g}] as in *king* [kɪŋ^{g}].

6.2 Error Patterns

In this section we give error patterns (which can be called pronunciation variations or error rules as well) which may operate in English speech generated by a L2 English learner whose mother tongue is Mexican Spanish. We built the rules given below based on a comparative analysis of American English and Mexican Spanish phonemes and allophones. This analysis is theoretic; in future we plan to validate these rules empirically by means of a comparative phonetic analysis of words/texts read by American English native speakers and L2 English learners with L1 Mexican Spanish.

Errors can be made due to similarities and/or differences of the spelling rules of two languages. In this work we considered only phonetic aspects without taking into account orthographic stereotypes of MS learners of AE.

The rules are given in two subsections, one for the vowels and the other for the consonants. The rules are represented according to the following patterns: on the left-hand side of a rule an AE sound is given; then, on the right-hand side of a rule, after an arrow, the MS sounds are given which may substitute the AE sound in English pronunciation of an MS speaker. If there is more than one MS sound which can substitute an AE sound, then such MS sounds are separated by a vertical bar (|). If two or more MS sounds are used to substitute an AE allophone, these MS sounds are combined using a plus (+) symbol. If in the latter combination of MS sounds one or more sounds can vary, the variation are separated by a vertical bar (|).

In some cases, an AE sound on the left-hand side of a rule looks exactly the same as an MS sound on the ride-hand side of the rule; for example, “Voiced bilabial nasal /m/ → voiced bilabial nasal /m/” (rule No. 22 in Section 6.2.2). Speaking in practical terms of acceptable pronunciation, it can be said, than the /m/ sound is pronounced correctly by an MS speaker. However, the MS /m/ is not exactly the same as the AE /m/, since the overall position of the speech organs are different in AE and MS. In spite of that, for teaching purposes, the AE /m/ can be considered the same as the MS /m/.

6.2.1 Vowels

1. High-front tense unrounded /i/ → high-front /i/
   - diphthongized [iɪ] → high-front /i/
   - diphthongized [iə] → high-front /i/ + mid-front /e/
   - reduced [ə] or [ɪ] → mid-front /e/ or high-front /i/
   - lengthened [iː] → high-front /i/
   - semi-lengthened [iˑ] → high-front /i/
   - shortened [i] → high-front /i/
2. Lower high-front lax unrounded /ɪ/ → high-front /i/
   - reduced [ə] → mid-front /e/
   - lengthened [ɪː] → high-front /i/
3. Mid-front tense unrounded /e/ → mid-front /e/
   - diphthongized [eɪ] → mid-front /e/ + high-front /i/
   - diphthongized and lengthened [eːɪ] → mid-front /e/ + high-front /i/
   - diphthongized and semi-lengthened [eˑɪ] → mid-front /e/ + high-front /i/
   - diphthongized and shortened [eɪ] → mid-front /e/ + high-front /i/
   - [i] or [ɪ] → high-front /i/
4. Lower mid-front lax unrounded /ɛ/ → mid-front /e/
   - diphthongized, r-colored and lengthened [ɛːɚ̯] → mid-front /e/ + low-central /a/ + sibilant flap [ɾ]
   - diphthongized, r-colored and semi-lengthened [ɛˑɚ̯] → mid-front /e/ + low-central /a/ + sibilant flap [ɾ]
   - diphthongized, r-colored and shortened [ɛɚ̯] → mid-front /e/ + low-central /a/ + sibilant flap [ɾ]
   - triphthongized [eɪə] → mid-front /e/ + high-front /i/ + mid-front /e/
5. Low-front lax unrounded /æ/ → mid-front /e/ | low-central /a/
   - lengthened [æ] → mid-front /e/ | low-central /a/
6. Lower mid-to-back central lax unrounded /ʌ/ → low-central /a/
   - [ɛ] → mid-front /e/
   - [ɪ] → high-front /i/
7. Neutral mid-central lax unstressed unrounded /ə/ → mid-front /e/
   - lower high-front lax unrounded [ɪ] → high-front /i/
8. Mid-central r-colored tense /ɝ/ → mid-front /e/ + sibilant flap [ɾ]
   - lengthened [ɝː] → mid-front /e/ + sibilant flap [ɾ]
   - semi-lengthened [ɝˑ] → mid-front /e/ + sibilant flap [ɾ]
   - shortened [ɝ] → mid-front /e/ + sibilant flap [ɾ]
9. Mid-central r-colored lax /ɚ̯/ → mid-front /e/ + sibilant flap [ɾ]
   - r-dropped [ə] → mid-front /e/
10. High-back tense rounded close /u/ → high-back /u/ | velar semi-vowel [u̯]
    - diphthongized [uə] → high-back /u/ + mid-front /e/
    - diphthongized [uʊ] → high-back /u/ | velar semi-vowel [u̯]
    - reduced [ʊ] or [ə] → high-back /u/ | velar semi-vowel [u̯] or mid-front /e/

   - lengthened [uː] → high-back /u/
   - semi-lengthened [uˑ] → high-back /u/
   - shortened [u] → high-back /u/ | velar semi-vowel [u̯]
11. High-back lax rounded /ʊ/ → high-back /u/ | velar semi-vowel [u̯]
   - reduced [ʌ] or [ə] → low-central /a/ or mid-front /e/
12. Mid-back tense rounded close /o/ → mid-back /o/
   - diphthongized [oʊ] → mid-back /o/ + velar semi-vowel [u̯] | high-back /u/
   - reduced [ə] → low-central /a/ | mid-front /e/
   - diphthongized and lengthened [oːʊ] → mid-back /o/ + velar semi-vowel [u̯] | high-back /u/
   - diphthongized and semi-lengthened [oˑʊ] → mid-back /o/ + velar semi-vowel [u̯] | high-back /u/
   - diphthongized and shortened [oʊ] → mid-back /o/ + velar semi-vowel [u̯] | high-back /u/
13. Low mid-back lax rounded open /ɔ/ → mid-back /o/
   - lengthened [ɔː] → mid-back /o/
   - semi-lengthened [ɔˑ] → mid-back /o/
   - shortened [ɔ] → mid-back /o/
   - lowered [ɒ] or [ɑ] → mid-back /o/ or low-central /a/
14. Low-back lax unrounded open /ɑ/ → mid-back /o/ | low-central /a/
   - rounded [ɒ] → mid-back /o/ | low-central /a/
   - fronted [a] → low-central /a/
   - fronted and rounded [ɔ] → mid-back /o/
15. Rising low-front to high-front diphthong /aɪ/ → low-central /a/ + high-front /i/
   - triphthongized [aɪə] → low-central /a/ + high-front /i/ + mid-front /e/
   - reduced [ə] → mid-front /e/
   - lengthened [aːɪ] → low-central /a/ + high-front /i/
   - semi-lengthened [aˑɪ] → low-central /a/ + high-front /i/
   - shortened [aɪ] → low-central /a/ + high-front /i/
   - elevated [ɜɪ] → mid-front /e/ + high-front /i/
16. Rising low-front to high-back diphthong /aʊ/ → low-central /a/ + velar semi-vowel [u̯] | high-back /u/
   - reduced [ʌʊ] → low-central /a/ + velar semi-vowel [u̯] | high-back /u/
17. Rising mid-back to high-front diphthong /ɔɪ/ → mid-back /o/ + high-front /i/
   - lengthened [ɔːɪ] → mid-back /o/ + high-front /i/
   - semi-lengthened [ɔˑɪ] → mid-back /o/ + high-front /i/
   - shortened [ɔɪ] → mid-back /o/ + high-front /i/

6.2.2 Consonants

1. Voiceless bilabial stop /p/ → voiceless bilabial unaspirated stop /p/

- /p/ with aspirated release [pʰ] → voiceless bilabial unaspirated stop /p/
- /p/ with unaspirated release [p̼] → voiceless bilabial unaspirated stop /p/
- /p/ with nasal release [p̃] → voiceless bilabial unaspirated stop /p/ + (optional: a reduced vowel similar to the MS vowel used to read the respective vowel letter following [p̃] in a given word or word combination, if any, otherwise a reduced vowel similar to the MS vowels /e/ or /a/)
- unreleased [p⁻] → voiceless bilabial unaspirated stop /p/ | omitted [ø]
- lengthened [p:] → voiceless bilabial unaspirated stop /p/
- preglottalized [ʔp] → voiceless bilabial unaspirated stop /p/

2. Voiced bilabial stop /b/ → voiced bilabial stop /b/ | approximant (spirantized) [β̞]
   - /b/ with nasal release [b̃] → voiced bilabial stop /b/ | approximant (spirantized) [β̞] + (optional: a reduced vowel similar to the MS vowel used to read the respective vowel letter following [b̃] in a given word or word combination, if any, otherwise a reduced vowel similar to the MS vowels /e/ or /a/)
   - unreleased [b⁻] → voiced bilabial stop /b/ | omitted [ø]
   - lengthened [b:] → voiced bilabial stop /b/ | approximant (spirantized) [β̞]
3. Voiceless alveolar stop /t/ → voiceless dental unaspirated stop /t/
   - /t/ with aspirated release [tʰ] → voiceless dental unaspirated stop /t/
   - /t/ with unaspirated release [t̼] → voiceless dental unaspirated stop /t/
   - /t/ with nasal release [t̃] → voiceless dental unaspirated stop /t/ + (optional: a reduced vowel similar to the MS vowel used to read the respective vowel letter following [t̃] in a given word or word combination, if any, otherwise a reduced vowel similar to the MS vowels /e/ or /a/)
   - unreleased [t⁻] → voiceless dental unaspirated stop /t/ | omitted [ø]
   - lengthened [t:] → voiceless dental unaspirated stop /t/
   - dentalized [t̪] → voiceless dental unaspirated stop /t/
   - flapped [ɾ] → voiceless dental unaspirated stop /t/ | voiced dental stop /d/ | approximant (spirantized) [ð̞] | sibilant flap [ɾ]
   - preglottalized [ʔt] → voiceless dental unaspirated stop /t/
   - glottal stop [ʔ] → voiceless dental unaspirated stop /t/ + (optional: a reduced vowel similar to the MS vowel used to read the respective vowel letter following [ʔ] in a given word or word combination, if any, otherwise a reduced vowel similar to the MS vowels /e/ or /a/)
   - affricated (palatalized) [tʃr̝̊] → voiceless dental unaspirated stop /t/ + devoiced hushing sibilant [r̝̊ˠ]
   - affricated (palatalized) [tʃ] → voiceless palatal affricate /c͡ʃ/ | voiceless dental unaspirated stop /t/ | voiceless dental unaspirated

stop /t/ + voiced dorsal palatal fricative /j/ | palatalized [ʒ] (allophone of voiceless dorosalveolar fricative /s/)

4. Voiced alveolar stop /d/ → voiced dental stop /d/ | approximant (spirantized) [ð̟]
   - /d/ with bilateral release [d‿l] → voiced dental stop /d/ | approximant (spirantized) [ð̟] + (optional: a reduced vowel similar to the MS vowel used to read the respective vowel letter following [d] in a given word or word combination, if any, otherwise a reduced vowel similar to the MS vowels /e/ or /a/) + voiced alveolar lateral liquid approximant /l/
   - /d/ with nasal release [d̃] → voiced dental stop /d/ | approximant (spirantized) [ð̟] + (optional: a reduced vowel similar to the MS vowel used to read the respective vowel letter following [d̃] in a given word or word combination, if any, otherwise a reduced vowel similar to the MS vowels /e/ or /a/)
   - unreleased [d⁻] → voiced dental stop /d/ | approximant (spirantized) [ð̟] | omitted [ø]
   - lengthened [d:] → voiced dental stop /d/ | approximant (spirantized) [ð̟]
   - dentalized [d̪] → voiced dental stop /d/ | approximant (spirantized) [ð̟]
   - flapped [ɾ] → voiced dental stop /d/ | approximant (spirantized) [ð̟] | sibilant flap [ɾ]
   - affricated (palatalized) [dʒr] → voiced dental stop /d/ + palatalized [ʒ] (allophone of voiceless dorosalveolar fricative /s/) + sibilant flap [ɾ]
   - affricated (palatalized) [dʒ] → voiced dental stop /d/ + palatalized [ʒ] (allophone of voiceless dorosalveolar fricative /s/)
5. Voiceless velar stop /k/ → voiced velar unaspirated stop /k/ | palatalized [kʲ]
   - /k/ with aspirated release [kʰ] → voiced velar unaspirated stop /k/ | palatalized [kʲ]
   - /k/ with unaspirated release [k̚] → voiced velar unaspirated stop /k/ | palatalized [kʲ]
   - /k/ with bilateral release [k‿l] → voiced velar unaspirated stop /k/ + (optional: a reduced vowel similar to the MS vowel used to read the respective vowel letter following [k] in a given word or word combination, if any, otherwise a reduced vowel similar to the MS vowels /e/ or /a/) + voiced alveolar lateral liquid approximant /l/
   - /k/ with nasal release [k̃] → voiced velar unaspirated stop /k/ + (optional: a reduced vowel similar to the MS vowel used to read the respective vowel letter following [k] in a given word or word combination, if any, otherwise a reduced vowel similar to the MS vowels /e/ or /a/)
   - unreleased [k⁻] → voiced velar unaspirated stop /k/ | omitted [ø]
   - lengthened [k:] → voiced velar unaspirated stop /k/ | palatalized [kʲ]

- preglottalized [ʔk] → voiced velar unaspirated stop /k/ | palatalized [kʲ]
- glottal stop [ʔ] → voiced velar unaspirated stop /k/ | omitted [ø]

6. Voiced velar stop /g/ → voiced velar stop /g/ | approximant (spirantized) [ɣ̞]
   - /g/ with bilateral release [g◡l] → voiced velar stop /g/ | approximant (spirantized) [ɣ̞] + (optional: a reduced vowel similar to the MS vowel used to read the respective vowel letter following [g] in a given word or word combination, if any, otherwise a reduced vowel similar to the MS vowels /e/ or /a/) + voiced alveolar lateral liquid approximant /l/
   - /g/ with nasal release [g̃] → voiced velar stop /g/ | approximant (spirantized) [ɣ̞] + (optional: a reduced vowel similar to the MS vowel used to read the respective vowel letter following [g̃] in a given word or word combination, if any, otherwise a reduced vowel similar to the MS vowels /e/ or /a/)
   - unreleased [g⁻]→ voiced velar stop /g/ | approximant (spirantized) [ɣ̞] | omitted [ø]
   - lengthened [gː] → voiced velar stop /g/ | approximant (spirantized) [ɣ̞]
7. Voiceless labiodental fricative /f/ → voiceless bilabial fricative /f/
   - interdental [θ] → voiceless bilabial fricative /f/ | voiceless dental unaspirated stop /t/
   - bilabial [ɸ] → voiceless bilabial fricative /f/
8. Voiced labiodental fricative /v/ → voiced bilabial stop /b/ | approximant (spirantized) [β̞]
   - devoiced [v̥] → voiceless bilabial fricative /f/
9. Voiceless interdental fricative /θ/ → voiceless dental unaspirated stop /t/ | voiceless bilabial fricative /f/
   - voiced [ð] → approximant (spirantized) [ð̞] | voiced dental stop /d/
10. Voiced interdental fricative /ð/ → approximant (spirantized) [ð̞] | voiced dental stop /d/
    - devoiced [ð̥] → voiceless dental unaspirated stop /t/ | voiceless bilabial fricative /f/
11. Voiceless alveolar fricative /s/ → voiceless dorosalveolar fricative /s/ | voiceless dental fricative /s̪/
    - palatalized [ʃ] → voiceless palatal fricative /ʃ/
12. Voiced alveolar fricative /z/ → voiceless dorosalveolar fricative /s/ | voiceless dental fricative /s̪/ | palatalized [ʒ] (allophone of voiceless dorosalveolar fricative /s/) | voiced [z] (allophone of voiceless dorosalveolar fricative /s/)
    - devoiced [z̥] → voiceless dorosalveolar fricative /s/ | voiceless dental fricative /s̪/
    - palatalized [ʒ] → palatalized [ʒ] (allophone of voiceless dorosalveolar fricative /s/)
    - stopping [d] → voiced dental stop /d/ | approximant (spirantized) [ð̞]

13. Voiceless palatal fricative /ʃ/ → voiceless palatal fricative /ʃ/
14. Voiced palatal fricative /ʒ/ → palatalized [ʒ] (allophone of voiceless dorosalveolar fricative /s/)
    - affricate [dʒ] → voiced dental stop /d/ + palatalized [ʒ] (allophone of voiceless dorosalveolar fricative /s/)
15. Voiceless glottal fricative /h/ → voiceless velar fricative /x/
    - voiced [ɦ] → approximant (spirantized) [ɣ̞] | voiceless velar fricative /x/
    - palatalized [ç] → voiceless velar fricative /x/ + voiceless palatal fricative /ʃ/ | voiced dorsal palatal fricative /j/
    - /h/ with glottal release [ʔ] → voiceless velar fricative /x/
    - omitted [ø] → omitted [ø]
16. Voiceless alveo-palatal affricate /tʃ/ → voiceless dental unaspirated stop /t/ + voiceless palatal fricative /ʃ/ (i.e., a combination of /t/ and /ʃ/) or voiceless palatal affricate /t͡ʃ/ (i.e., a single sound /t͡ʃ/)
17. Voiced alveo-palatal affricate /dʒ/ → voiced dental stop /d/ | approximant (spirantized) [ð̞] + palatalized [ʒ] (allophone of voiceless dorosalveolar fricative /s/)
18. Voiced labiovelar glide approximant /w/ → velar semi-consonant [w] (allophone of high-back /u/)
    - aspirated [hw] → voiceless velar fricative /x/ | omitted [ø] + velar semi-consonant [w] (allophone of high-back /u/)
    - devoiced [w̥] → velar semi-consonant [w] (allophone of high-back /u/)
19. Voiced alveopalatal liquid approximant /r/ → voiced alveolar thrill approximant /r/ | sibilant flap [ɾ]
    - devoiced [r̥] → voiced alveolar thrill approximant /r/ | sibilant flap [ɾ] | devoiced hushing sibilant [r̥ˢ]
    - flap [ɾ] → sibilant flap [ɾ]
    - retroflexed [ɻ] → voiced alveolar thrill approximant /r/ | sibilant flap [ɾ]
    - back [r̟] → voiced alveolar thrill approximant /r/ | sibilant flap [ɾ]
20. Voiced palatal glide approximant /j/ → voiced dorsal palatal fricative /ʝ/
    - omitted [ø] → omitted [ø] | voiced dorsal palatal fricative /ʝ/
    - devoiced [j̥] → voiced dorsal palatal fricative /ʝ/ | voiceless palatal fricative /ʃ̊/
21. Voiced alveolar lateral liquid approximant /l/ → voiced alveolar lateral liquid approximant /l/
    - light [l] → voiced alveolar lateral liquid approximant /l/
    - dark, velarized [ɫ] → voiced alveolar lateral liquid approximant /l/
    - syllabic, also dark [l̩] → voiced alveolar lateral liquid approximant /l/
    - devoiced [l̥] → voiced alveolar lateral liquid approximant /l/
    - dentalized [l̪̊] → voiced alveolar lateral liquid approximant /l/
22. Voiced bilabial nasal /m/ → voiced bilabial nasal /m/
    - syllabic [m̩] → voiced bilabial nasal /m/

- lengthened [m:] → voiced bilabial nasal /m/
- labiodentalized [m̪] → voiced bilabial nasal /m/

23. Voiced alveolar nasal /n/ → voiced alveolar nasal /n/
    - syllabic [n̩] → voiced alveolar nasal /n/
    - lengthened [n:] → voiced alveolar nasal /n/
    - labildentalized [n̪] → dentalized [n̪]
    - dentalized [n̪] → dentalized [n̪]
    - velarized [n̩] → velarized [n̩]
24. Voiced velar nasal /ŋ/ → velarized [n̩] (allophone of voiced alveolar nasal /n/)
    - syllabic [n̩] → velarized [n̩] (allophone of voiced alveolar nasal /n/)
    - alveolarized [n] → voiced alveolar nasal /n/
    - stop [$ŋ^k$] or [$ŋ^g$] → voiced alveolar nasal /n/ | velarized [n̩] + voiced velar stop /g/ | approximant (spirantized) [ɣ̞] | voiceless velar fricative /x/

7 Conclusions and Future Work

In this article, we presented error patterns built on the basis of our comparative analysis of American English and Mexican Spanish phonemes and allophones. These error patterns (or error rules) can be applied in designing the error detection module of a Computer Assisted Pronunciation Training (CAPT) System for teaching American English pronunciation to Mexican Spanish speakers.

Since individual error detection or localization for any language in general is a very difficult computational task in the area of automatic speech recognition, our error rules can help to improve the precision of error identification in intelligent tutor systems for teaching American English pronunciation.

To the best of our knowledge, error patterns in American English speech generated by Mexican Spanish speakers has not been defined in previous work which was done mainly for Castilian-originated standard Spanish. Moreover, the state of the art analysis was done for phonemes, and in this work we performed our analysis for the most common allophones of the phonemes. It is a significant contribution to the field, as allophones, not phonemes, are sounds generated in real-life speech, and a good mastering of allophones is what produces a less accented speech of an L2 English learner.

Also, error patterns can be implemented in automatic less-accented speech generation as well as in automatic error correction systems.

In future, we plan to empirically verify the theoretically derived error patterns in this work on the material of an English learner corpus including speech generated by Mexican Spanish native speakers.

References

1. Avery, P., Ehrlich, S.: Teaching American English Pronunciation. Oxford University Press, England (1992)

2. Burbules, N. C.: Ubiquitous Learning and the Future of Teaching. Encounters on Education, vol. 13, pp. 3–14 (2012)
3. Cruttenden, A.: Gimson's pronunciation of English. The 8th edition. Routledge, New York (2014)
4. DeMori, R., Suen, C. Y.: New Systems and Architectures for Automatic Speech Recognition and Synthesis. Springer-Verlag NY Inc. (2012)
5. Edwards, H. T.: Applied Phonetics: the Sounds of American English. Singular Pub. Group, San Diego, CA (1997)
6. Eskenazi, M.: An overview of spoken language technology for education. Speech Communication, vol. 51(10), pp. 832–844 (2009)
7. Granger, S.: Learner English on Computer. Routledge, New York (2014)
8. Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet. Cambridge University Press, UK (1999)
9. Hismanoğlu, M.: The integration of information and communication technology into current ELT coursebooks: a critical analysis. Procedia-Social and Behavioral Sciences, vol. 15, pp. 37–45 (2011)
10. Ito, A., Lim, Y. L., Suzuki, M. & Makino, S.: Pronunciation error detection method based on error rule clustering using a decision tree. In Ninth European Conference on Speech Communication and Technology (2005)
11. Khan, B. H.: A Comprehensive E-Learning Model. Journal of e-Learning and Knowledge Society, vol. 1, pp. 33–43 (2005)
12. Levy, M., Stockwell, G.: CALL Dimensions: Options and Issues in Computer-Assisted Language Learning. Lawrence Erlbaum Associates, Inc., NJ (2006)
13. Liakin, D.: Mobile-Assisted Learning in the Second Language Classroom. International Journal of Information Technology & Computer Science, vol. 8(2), pp. 58–65 (2013)
14. Menzel, W., Herron, D., Bonaventura, P., Morton, R.: Automatic detection and correction of non-native English pronunciations. Proceedings of INSTILL, pp. 49–56 (2000)
15. Moreno de Alba, J. G.: El español en América. Fondo de cultura económica, México (2001)
16. Neri, A., Cucchiarini, C., Strik, W.: Automatic speech recognition for second language learning: how and why it actually works. In: Proceedings of ICPhS, pp. 1157–1160 (2003)
17. Pieraccini, R.: The voice in the machine: building computers that understand speech. MIT Press (2012)
18. Pineda, L.A., Castellanos, H., Cuétara, J., Galescu, L., Juárez, J., Llisterri, L., Pérez, P., Villaseñor, L.: The Corpus DIMEx100: Transcription and Evaluation. Language Resources and Evaluation, vol. 44(4), pp. 347–370 (2010)
19. Quilis, A.: El comentario fonológico y fonético de textos: teoría y práctica. 3a edición. Arco/Libros, S.L., Madrid (1997)
20. Strik, H., Truong, K., de Wet, F., Cucchiarini, C.: Comparing different approaches for automatic pronunciation error detection. Speech Communication, vol. 51(10), pp. 845–852, doi: 10.1016/j.specom.2009.05.007 (2009)
21. Truong, K., Neri, A., Cucchiarini, C. & Strik, H.: Automatic pronunciation error detection: an acoustic-phonetic approach. In InSTIL/ICALL Symposium (2004)

22. Weigelt, L. F., Sadoff, S. J. & Miller, J. D.: Plosive/fricative distinction: The voiceless case. The Journal of the Acoustical Society of America, vol. 87(6), pp. 2729–2737 (1990)
23. Whitley, M.S.: Spanish-English Contrasts: A Course in Spanish linguistics. Georgetown University Press, Washington, D.C. (1986)
24. Witt, S.: Use of speech recognition in Computer assisted Language Learning. PhD thesis. Department of Engineering, University of Cambridge, UK (1999)
25. Yoon, S. Y., Hasegawa-Johnson, M. & Sproat, R.: Landmark-based automated pronunciation error detection. In Interspeech, pp. 614–617 (2010)
26. Zhao, T., Hoshino, A., Suzuki, M., Minematsu, N. & Hirose, K.: Automatic Chinese pronunciation error detection using SVM trained with structural features. In SLT, pp. 473–478 (2012)

Enriquecimiento automático de un léxico afectivo basado en relaciones semánticas obtenidas de un diccionario explicativo en español

Noé Alejandro Castro-Sánchez y Bernardo López-Santiago

Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), Cuernavaca, Morelos, México

{ncastro, bernals}@cenidet.edu.mx

Resumen. Las relaciones semánticas de sinonimia e hiperonimia son estudiadas y utilizadas en varios problemas de PLN. En este trabajo se estudian y procesan para incrementar las palabras en un léxico afectivo, con base en la determinación de patrones que utiliza el diccionario de la Real Academia de la Lengua Española en las definiciones de sustantivos y verbos. La definición de los diccionarios tiene patrones que permiten, a partir de heurísticas, extraer relaciones de sinonimia e inclusión. En este trabajo se desarrollaron dichas heurísticas y se comprobó que obtienen los sinónimos e hiperónimos de las definiciones a un bajo costo computacional, logrando agregar poco más de 1300 palabras al léxico.

Palabras clave: Relaciones semánticas, sinonimia, hiperonimia, definiciones lexicográficas, léxico afectivo.

1. Introducción

Las relaciones semánticas se han estudiado y utilizado ampliamente en diversas tareas de Procesamiento de Lenguaje Natural (PLN). Podemos dividir el estudio de estas relaciones según la manera en que se presenta el texto en los documentos procesados, esto es, como estructurado (por ejemplo diccionarios) y no estructurado (por ejemplo páginas web).

Las ventajas de utilizar texto estructurado es que se requiere una cantidad menor de documentos en comparación con el texto no estructurado, lo que repercute que su procesamiento tenga un bajo costo computacional. El hecho de que esté estructurado, significa que presenta una regularidad estructural y sintáctica, que permite a los métodos basados en reglas obtener buenos resultados.

El descubrimiento de hiperónimos con texto no estructurado generalmente necesita de un gran corpus (hablamos de millones de documentos). En el idioma inglés se han

realizado trabajos para descubrir hiperónimos utilizando los llamados "Patrones de Hearts" [1]. Por otro lado, el descubrimiento de sinónimos en este tipo de texto no es una tarea trivial, ya que las relaciones de sinonimia se establecen con mayor frecuencia por la semántica y no por la sintaxis. Se opta por un enfoque distributivo para el descubrimiento de sinónimos en texto no estructurado [2].

En este artículo nos centraremos en las definiciones utilizadas en el Diccionario de la Real Academia Española (DRAE) basado en el análisis de la estructura de palabras dentro de la categoría de verbos y sustantivos, esto con el fin de encontrar patrones que nos faciliten la identificación de relaciones semánticas de sinonimia e inclusión y así poder realizar el enriquecimiento de un léxico afectivo.

2. Léxico afectivo utilizado

Los indicadores más importantes de sentimientos son *sentiment words*, también llamadas *opinion words*. Estas palabras se utilizan comúnmente para expresar sentimientos positivos o negativos. Por ejemplo: bueno *(good)*, maravilloso *(wonderful)* y asombroso *(amazing)* son palabras que expresan sentimientos positivos, y malo *(bad)*, escaso (*poor)* y terrible *(terrible)* son palabras que expresan sentimientos negativos. Una lista de esas palabras es llamada léxico afectivo [3].

Un léxico afectivo puede estar divido en ciertas clases de sentimientos, por ejemplo: positivo/negativo o en una única lista de palabras asociadas a un valor numérico que representa su polaridad [4].

En este trabajo se procesa el léxico afectivo en español (llamado LAfE) [5] el cual se realizó a partir de una traducción manual de recursos como léxicos afectivos ya existentes en el idioma inglés (General Inquirer [6], Opinion Finder [7], WordNetAffect [8]) y de teorías psicológicas (Geneva Emotion Wheel [9], Geneva Affect Label Coder [10], A Circumplex Model of Affect [11], Structure of Emotions [12]).

En su fase de creación, el léxico afectivo se enriqueció de manera manual a través de relaciones de sinonimia, inclusión y familia léxica de las palabras que se localizaron en los léxicos en inglés.

El léxico afectivo se compone de un total de 3,325 palabras, de las cuales 1,178 son sustantivos, 1,275 verbos, 820 adjetivos y 50 adverbios.

Las palabras contenidas en el léxico se encuentran etiquetadas en cuatro dimensiones de polaridad: positiva, muy positiva, negativa y muy negativa. En algunas teorías psicológicas y en WordNetAffect, algunas palabras se encuentran asociadas a emociones, en esos casos se agregó la palabra que designa la emoción asociada. Se registró el recurso de dónde se extrajo cada palabra y las palabras en inglés de las que provienen.

En la Tabla 1 se muestra un extracto del contenido de dicho léxico. Los léxicos afectivos se indican como: GI que referencia a *General Inquirer*, WNA a *WordNetAffect* y OF a *Opinion Finder*. Por otro lado, las teorías psicológicas se indican con el nombre de sus autores: Morgan-Heise, Rusell y Scherer. La simbología para representar la polaridad es la siguiente: ++ hace referencia a una polaridad muy

positiva, + indica polaridad positiva, - se asocia a una polaridad negativa y finalmente – hace referencia a muy negativa.

Table 1. Estructura del léxico afectivo en español.

Palabra	*Polaridad*	*Emoción*	*Morgan-Heise*	*Rusell*	*Scherer*	*GI*	*WNA*	*OF*	*inglés*
Regocijo	+	Alegría				+			rejoice
Angustiado	--	Ansiedad	--						anguish
Tranquilo	+	Serenidad		+					calm
Deplorable	-	Tristeza					-		deplorable
Despojado	-	Tristeza					-		bereft
Abatido	--	Tristeza			--				dejected
Furioso	--	Ira	--						furious
Furiosamente	--	Ira							
Colérico	--	Ira							
Solidaridad	+	Ninguna						+	solidarity

3. Definiciones en el diccionario de la RAE

3.1. Estructura de los artículos del diccionario

Las secciones textuales dispuestas ordenadamente en un diccionario se denominan artículos, y están conformadas por una entrada también denominada unidad léxica, y la información que la define o describe. Además de estos dos elementos, se ha llegado también a considerar la categoría gramatical de la entrada como parte del artículo.

Las entradas pueden ser simples (una sola palabra) o complejas (más de una palabra), y aparecen ordenadas alfabéticamente en el diccionario en su forma lematizada.

Delante de la unidad léxica se disponen información relativa de ella, de la cual puede distinguirse una serie de elementos que señalan sus restricciones y condiciones de uso, y la información semántica, o definición, que constituye el contenido básico del artículo lexicográfico.

3.2. Tipos de definiciones en el diccionario

El principio para que una definición se considere correcta es que ésta debe abarcar todo lo definido pero nada más que lo definido, y una forma de comprobar si se cumple esta condición es aplicando la prueba de la *sustituibilidad* [13].

Se puede considerar de manera muy general una distinción entre dos tipos de definiciones: de contenido y de signo. En las definiciones de contenido o conceptuales, se considera el definido como una unidad que hace referencia a la realidad y se pretende traducir en otras palabras de la misma lengua su contenido significativo. En las definiciones de signo o funcionales, se considera al definido

como elemento o signo del sistema de la lengua, por lo tanto se informa de sus valores y funciones dentro del sistema.

La definición conceptual se divide en: definición perifrástica y definición sinonímica.

En la definición sinonímica se define una palabra en términos de otra que tiene el mismo (o cercano) significado que la que se desea definir [14,15].

La definición conceptual perifrástica puede manifestarse bajo diversas formas teniendo en cuenta la naturaleza de la unidad a definir. En [13] se menciona el siguiente esquema con los diferentes tipos de definición:

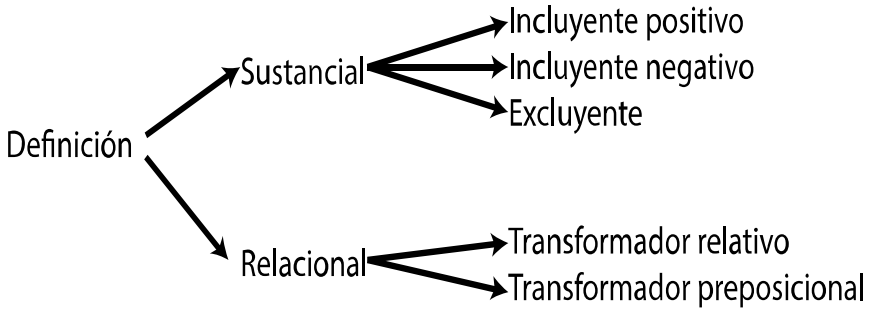

Fig. 1. Tipos de definición conceptual perifrástica.

La definición sustancial intenta responder a la pregunta « ¿Qué es el definido? ». La respuesta suele darse de las siguientes formas:

- el definido es «tal cosa» (incluyente positiva),
- el definido es «no tal cosa» (incluyente negativa),
- el definido «es contrario» o «carece de tal cosa» (excluyente).

La definición incluyente positiva viene a ser el prototipo de definición lógica aristotélica que analiza al definido mediante el género próximo (palabra cuya carga semántica abarca al definido, también se le conoce como hiperónimo) y la diferencia específica (encargada de concretar el significado del definido) [13][15].

La definición relacional remite a la relación entre el definido cualificante y otra palabra cualificada. Consta de dos partes: un elemento transformador, que puede ser un relativo (definición relativa) o una preposición (definición preposicional), el cual confiere a la palabra u oración que sirve de definidor (elemento traspuesto) el carácter de adjetivo o adverbio [13].

Dejando los modelos formales de definición cabe destacar la llamada definición morfo-semántica [13]. La definición morfo-semántica es utilizada en la definición de palabras compuestas y derivadas, la definición viene a resultar en la descomposición de los elementos componentes. Relacionadas con este grupo se encuentran los sustantivos derivados formados a partir de un verbo con la intervención de distintos sufijos nominales o de derivados adjetivos a partir de sustantivos.

4. Metodología de solución

4.1. Etiquetas de vigencia de uso y voces técnicas en el diccionario

Como primer filtro para encontrar las relaciones de sinonimia se discriminaron las entradas con las etiquetas de vigencia de uso (ant., desus., p. us.) y voces técnicas [16]. La marcación sobre la vigencia de la palabra aparece cuando se pretende informar sobre su bajo empleo [13][17].

Analizando las muestras notamos que las palabras con etiquetas de vigencia ya no se utilizan de la misma forma en el español actual y que comúnmente tienen un significado diferente, por ejemplo en la palabra "**enojo**" en su acepción tercera nos remite a la definición de "**agravio**" y más particularmente a la acepción de ofensa, que actualmente en ningún contexto se utilizan como sinónimos, lo mismo sucede con definiciones con la marca "desus." y "p.us.", como se muestra a continuación:

***Enojo: 3**. m. ant. agravio (||ofensa).*
***Calma: 6**. f. desus. Angustia, pena.*
***Aprobación: 2**. f. desus. prueba.*
***Agrado: 3**. m. Ec. p. us. obsequio (|| regalo).*

También se observó que las palabras marcadas con voces técnicas no son relevantes para el estudio ya que no mostraban rasgos de afectividad. Por ejemplo:

Idiocia: 1. f. Med. Trastorno caracterizado por una deficiencia muy profunda de las facultades mentales, congénita o adquirida en las primeras edades de la vida.
Fuego: 11. m. Med. cauterio.
Deducción: 3. f. Fil. Método por el cual se procede lógicamente de lo universal a lo particular.
4. f. Mús. Serie de notas que ascienden o descienden diatónicamente o de tono en tono sucesivos.

4.2. Definiciones morfo-demánticas

Después de este filtrado se buscaron patrones en las definiciones llegando a los siguientes resultados: En el trabajo se identificaron sustantivos que para definirse hacen uso de la definición morfo-semántica utilizando la paráfrasis *"Acción y efecto de + verbo"* y *"Acción de + verbo"*. Las palabras obtenidas en estas definiciones pertenecen a la familia léxica del sustantivo. Por ejemplo:

*Desmoralización: 1. acción y efecto de **desmoralizar**.*
*Quebranto: 1. acción y efecto de **quebrantar** o **quebrantarse.***
*Arresto: 1. acción de **arrestar**.*
*Perdición: 1. acción de **perder** o **perderse**.*

Dentro de la misma categoría de definiciones morfo-semánticas, en la categoría de sustantivos se localizaron definiciones del tipo *"cualidad de"* y la palabra que la complementa puede ser un adjetivo, sustantivo o verbo. Estas palabras también forman parte de la familia léxica del sustantivo definido. Por ejemplo:

Fealdad: 1. cualidad de ***feo.***
Desigualdad: 1. cualidad de ***desigual.***
Ternura: 1. cualidad de ***tierno.***
Caballerosidad: 1. cualidad de ***caballeroso.***

4.3. Definiciones sinonímicas

Los patrones encontrados en la definición de tipo sinonímica (tanto para sustantivos como verbos) son los siguientes:

- Sustantivos separados por el signo coma (,). Por ejemplo:
  *Quebranto: Lástima**,** conmiseración**,** piedad.*
  *Vacilación: Perplejidad**,** irresolución.*
  *Equilibrio: contrapeso**,** contrarresto**,** armonía entre cosas diversas.*
- Sustantivos separados por una disyunción, por ejemplo:
  Flojera: Debilidad ***o*** *cansancio.*
  Remedio: Enmienda ***o*** *corrección.*
- Sustantivos separados por comas y una disyunción, por ejemplo:
  *Declinación: Caída**,** descenso* ***o*** *declive.*
  *Hiel: Amargura**,** aspereza* ***o*** *desabrimiento.*

4.4. Definiciones de la lógica aristotélica (relaciones de tipo hipónimo-hiperónimo)

En [13] el autor hace distinción de los tipos de definiciones que se utilizan en el diccionario e identifica las más utilizadas para definir las palabras que caen en la categoría gramatical de sustantivos, así argumenta que los sustantivos utilizan comúnmente la definición conceptual perifrástico de tipo sustancial en cualquier modalidad. Es muy común que en este tipo de definiciones el hiperónimo de la palabra definida sea la primera palabra de la definición. Este patrón se observó en muchas de las definiciones de los sustantivos, por ejemplo en la definición de "llanto", podemos nota que su hiperónimo es "efusión", el llanto es un tipo de efusión. Algunos ejemplos son:

Efusión: Derramamiento de un líquido, y más comúnmente de la sangre.
Llanto: ***Efusión*** *de lágrimas acompañada frecuentemente de lamentos y sollozos.*

Aversión: Rechazo o repugnancia frente a alguien o algo
Horror: ***Aversión*** *profunda hacia alguien o algo.*

5. Resultados

Extrayendo las palabras de las definiciones morfo-semánticas se pudieron agregar 125 palabras nuevas al léxico afectivo, dichas palabras pertenecen a la familia léxica de los sustantivos extraídos del mismo léxico procesado.

Para agregar las nuevas palabras encontradas por sinonimia se verificó que al menos una palabra de la acepción de la definición procesada estuviera en el léxico afectivo, esto con el fin de garantizar que la palabra encontrada sea una palabra afectiva. Después de esta verificación se agregaron los sinónimos que aún no se encontraban en el léxico afectivo.

El método para la extracción de hiperónimos tuvo una precisión del 76%, la validación se realizó extrayendo 200 palabras de manera aleatoria de la lista de posibles hiperónimos obtenidos y buscando su definición en el diccionario para comprobar que el hiperónimo es correcto. Para determinar si es correcto el hiperónimo la definición del mismo debe abarcar al definido. Los hiperónimos nuevos encontrados para sustantivos y que se agregarían al léxico afectivo suman la cantidad de 749 palabras.

Las palabras agregadas al léxico conservaron la polaridad de la palabra que se utilizó en el léxico para extraer sus relaciones semánticas, es decir si utilizamos una palabra positiva, sus sinónimos, hiperónimos y familia léxica conservan esa polaridad positiva.

En la siguiente tabla se resumen los resultados obtenidos:

Table 2. Palabras nuevas agregadas al léxico afectivo.

Relación semántica	*Palabras agregadas al léxico de manera automática*
Sinonimia de sustantivos	167
Sinonimia de verbos	357
Hiperonimia	749
Familia léxica	125

5. Conclusiones

Podemos determinar parte de la familia léxica de los sustantivos y verbos de una forma relativamente fácil. La forma estructurada de las definiciones del diccionario hace idóneo el uso de patrones. Se obtienen buenos resultados en la identificación de sinonimia con un procesamiento relativamente sencillo y rápido, en comparación con los métodos basados en corpus donde además es necesaria una gran cantidad de documentos para obtener una precisión aceptable.

La polaridad en los hiperónimos no siempre se mantiene como habíamos supuesto en un principio: esto se debe a que nos encontramos hiperónimos que no tienen un sentido afectivo, es decir son entes abstractos, por ejemplo en la palabra “compasión” se tiene como hiperónimo a la unidad léxica “sentimiento” al cuál no se le puede

asignar alguna polaridad. Algunos de los hiperónimos que se encontraron con esta falta de sentido afectivo, son "acto", "estado", "cosa", "persona", "dispositivo", "sentido", "capacidad" y "sentimiento".

Además en la búsqueda de sinónimos nos encontramos con que algunas palabras tienen ambas polaridades y esto se debe a que en una acepción o bajo determinado contexto tienen una polaridad positiva, pero en otra acepción o contexto tienen una polaridad negativa. Por ejemplo la palabra "atrevimiento" es sinónimo de las siguientes palabras existentes en el léxico afectivo: insolencia (polaridad negativa), descaro (polaridad negativa), audacia (polaridad muy positiva) y osadía (polaridad muy positiva). Las palabras que presentaron esta característica (16 palabras), no fueron agregadas al léxico.

Bibliografía

1. Ritter, A., Soderland, S., & Etzioni, O.: What Is This, Anyway: Automatic Hypernym Discovery. AAAI Spring Symposium: Learning by Reading and Learning to Read, 88–93 (2009)
2. Wang, T., & Hirst, G.: Exploring patterns in dictionary definitions for synonym extraction. Natural Language Engineering, 18(03), 313–342 (2012)
3. Liu, B.: Sentiment Analysis and Opinion Mining. (G. Hirst, Ed.) Morgan & Claypool Publishers (2012)
4. Pak, A.: Automatic, adaptive and applicative Sentiment Analysis. Paris: Université Paris SUD. A thesis submitted in fulfillment of the requirements for the degree of Philosophy Doctor in Computer Science (2012)
5. Baca Gómez, R. Y.: Desarrollo de un servicio web para determinar la polaridad de textos de redes sociales en español. Cuernava, Morelos, Mexico: CENIDET (2014)
6. Stone, P., Dunphy, D., Smith, M., & Ogilvie, D.: The General Inquirer: A Computer Approach to Content Analysis. MIT Press (1966)
7. Wilson, T., Hoffman, P., Somasundaran, S., Kessler, J., Wiebe, J., Choi, Y., et al.: OpinionFinder: A system for subjectivity analysis. In Proceedings of HLT/EMNLP Interactive Demonstrations (2005)
8. Strapparava, C., & Valitutti, A.: WordNet-Affect: an Affective Extension of WordNet. In Proceedings of the 4th International Conference on Language Resources and Evaluation, 1083–1086 (2004)
9. Sacharin, V., Schlegel, K., & Scherer, K. R.: Geneva Emotion Wheel rating study (Report). Geneva, Switzerland: University of Geneva. Swiss Center for Affective Sciences (2012)
10. Scherer, K. R.: What are emotions? And how can they be measured? Social science information, 44(4), 695–729 (2005)
11. Russell, J.: A Circumplex Model of Affect. Journal of Personality and Social Psychology, 39(6), 1161–1178 (1980)
12. Morgan, R., & Heise, D.: Structure of Emotions. Social Psychology Quarterly, 51(1), 19–31 (1988)
13. Pérez Lagos, Manuel Fernando: Sobre algunos aspectos del quehacer lexicográfico. ELUA. Estudios de Lingüística. Vol. 12, pp. 163–179 (1998)

14. Wilson, T., Hoffman, P., Somasundaran, S., Kessler, J., Wiebe, J., Choi, Y., et al.: OpinionFinder: A system for subjectivity analysis. In Proceedings of HLT/EMNLP Interactive Demonstrations (2005)
15. Real Academia Española, «RAE» [En línea]. Available: http://www.rae.es/publicaciones/62-definiciones. [Último acceso: 20 Octubre 2014]
16. Real Academia Española, «RAE» [En línea]. Available: http://www.rae.es/diccionario-de-la-lengua-espanola/que-contiene/item-numero-2#_Toc85519269. [Último acceso: 21 Octubre 2014]
17. Real Academia Española, «RAE» [En línea]. Available: http://www.rae.es/diccionario-de-la-lengua-espanola/que-contiene/item-numero-2#_Toc85519266. [Último acceso: 21 Octubre 2014]

Reviewing Committee

Alexander Gelbukh
Felix Castro Espinoza
Francisco Viveros Jiménez
Grigori Sidorov
Gustavo Arroyo Figueroa
Hugo Terashima Marín
Ildar Batyrshin
Jesús González Bernal
Luis Villaseñor Pineda
Maya Carillo Ruiz
Miguel González Mendoza
Noé Alejandro Castro Sánchez
Obdulia Pichardo Lagunas
Omar Montaño Rivas
Oscar Herrera Alcantara
Rafael Murrieta Cid
Sabino Miranda Jiménez
Sofía Galicia Haro

Impreso en los Talleres Gráficos
de la Dirección de Publicaciones
del Instituto Politécnico Nacional
Tresguerras 27, Centro Histórico, México, D.F.
Noviembre de 2014
Printing 500 / Edición 500 ejemplares