A Knowledge-Based information Extraction Prototype for Data-Rich Documents in the Information Technology Domain

By

Sergio Gonzalo Jimenez Vargas
sgjimenezv@unal.edu.co

A dissertation submitted in partial fulfillment of the requirements for the degree of

Master of Sciences

Master in Systems Engineering and Computer Science
Systems and Industrial Engineering Department
National University of Colombia (Bogota D.C.)

2008

NATIONAL UNIVERSITY OF COLOMBIA

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Sergio Gonzalo Jimenez Vargas

This thesis has been read by each member of the following graduate committee and has been found to be satisfactory.

_____	_____
Date	Fabio A. Gonzalez O. (Chair)
_____	_____
Date	???? ????
_____	_____
Date	???? ????
_____	_____
Date	???? ????

A Knowledge-Based information Extraction Prototype for Data-Rich Documents in the Information Technology Domain

Sergio Gonzalo Jimenez Vargas

Abstract

The Internet is a generous source of information. Semi-structured text documents represent great part of that information; commercial data-sheets of the Information Technology domain are among them (e.g. laptop computer data-sheets). However our capacity to automatically gather and manipulate such information is limited due to the fact that those documents are designed to be read by people. Many documents in domains such as the one of Information Technology describe their commercial products in data-sheets with technical specifications. People use those data-sheets mainly to make buying decisions. Commercial data-sheets are a kind of *data-rich* documents. Data-rich documents have: a limited utilization of complex natural language structures, heterogeneous format, specialized terminology, names, abbreviations, acronyms, quantities, magnitudes and units.

This thesis presents an information extractor for *data-rich* documents based on the knowledge represented in a lexicalized domain ontology built mainly with meronymy and attribute relationships. Ontology concepts were manually lexicalized with words and n-word terms in English. The extraction process is mainly composed of a fuzzy string matcher module and a word sense disambiguation-inspired (WSD) module. The former was used to find references to ontology concepts allowing an error margin and the later was used to choose the better concept and use/sense associated with each referenced concept in a data-rich document according to its respective context.

A domain ontology, a lexicon and a labeled corpus were manually constructed for the laptop computers domain using data-sheets downloaded from the web sites of three of the most popular computer brands. The ontology had more than 350 concepts and the lexicon had more than 380 entries with at least 1500 different related terms. The validation corpus was five selected data-sheet documents with more than 5000 tokens in total of which 2300 are extraction targets. The built ontology and lexicon not only fitted the validation corpus but it covered a set of 30 laptop data-sheets.

The problem of approximate text string matching using static measures were reviewed. The static string measures are those that compares two strings in an algorithmic way using only the information contained in both strings. Additionally, a comparative study of different string comparison techniques at character and at token-level was made using data sets well known in the related literature. Particularly, a new general fuzzy cardinality-based combination method

of string measures at character level with resemblance coefficients (e.g. Jaccard, Dice, cosine) is proposed and encouraging results were obtained.

In order to develop the WSD-inspired disambiguator, the concept of *semantic path* was proposed allowing three basic uses: (i) to define an unambiguous label for semantic document annotation, (ii) to determine the use/sense inventory for each ontology concept and (iii) to serve as comparison unit for semantic relatedness measures. A *semantic path* is a chain of concepts that connects the ontology root concept with in a terminal concept (or "sink" node) in an ontology directed acyclic graph.

The proposed disambiguation technique was inspired in WSD methods based on general lexicalized ontologies such as WordNet. Additionally, a new global context graph-optimization criterion for disambiguation was proposed (i.e. shortest path). That criterion seemed to be suitable for the specific task reaching a F-measure above the 80%. Finally, the information extractor showed to be resilient against random noise introduced in the lexicon.

Acknowledgments

I am indebted to many people for their assistance and support to my thesis work. This work could not have been possible without the support of many people inside and outside of National University of Colombia. I would like to express particular appreciation to the following people.

To my wife, colleague and partner, Claudia Becerra, who has provided the main inspiration, encouragement and every day moral support in completing this thesis. I also appreciate her love, help and tolerance of all the late nights and weekends which we have put in studying, discussing and working, I am grateful.

To my advisor, Dr. Fabio Gonzalez, whose enthusiasm, intelligence and advice inspired me to accomplish this work. I would like to thank him for all the generous time in his very busy schedule to discuss ideas. I also would like to thank Dr. Alexander Gelbukh, for his valuable commentaries that reoriented our approach to the right way in the right time.

To Dr. Jonatan Gómez, who advised me during my efforts to find boundaries for this work and provide invaluable suggestions. To Dr. Yoan Pinzon, for his time and support in my technical writing for polishing my thesis. To Dr. Luis F. Niño and Dr. Elizabeth León, their skillful teaching and patient tutoring were helpful to me in the early stages of this project. To Zaza Aziz, who patiently help me to improve my English skills. Special thanks to my colleagues at the Laboratory of Intelligent Systems (LISI) and to the fellow students of the three research seminars who have provided their valuable commentaries and feedback during innumerable presentations, reviews and waiting times.

I also would like to thank Emilse Villamil, who has provided invaluable assistance taking kindly care of my daughters. Thanks for her help and patience with my irregular schedule and undoubtedly for her healthy and delicious food.

Finally, I would like to thank my daughters Sara and Luisa for their bright smiles, company, happiness and unconditional love that gave to me the necessary motivation to support all the time in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The amount of documents available electronically has increased dramatically with the vertiginous growth of the Web, but our capacity to "understand" automatically those documents is still considered an open problem [34]. On the other hand, *information extraction* (IE) is a natural language processing field that aims to extract structured information from unstructured text documents. For instance, extracting from a set of literary critics a list of authors, book titles and publishers is an IE task. IE is considered a shallow reading process but is a step towards solving the challenge of machine-reading.

Particularly, IE aims to find some selected entities and simple relations in text documents. In the book publishing example, three target fields can be extracted (*i.e.* author, book title and publisher) and two possible relations are "is-written-by" and "is-published-by". Usually, the targets of an IE process is only for small part of document information in comparison with the entire information contained in it. Many other entities and complex relations expressed in the document are disregarded. For instance, criticisms, opinions, feelings, sarcastic remarks, recommendations are out of the reach of the IE processes.

However, other document types can be considered in which the amount of information that can be extracted is comparable to the total document information. For instance a laptop data-sheet describes the product in detail, including features, composing parts and performance capabilities. Some examples of data-sheets are shown in figure 1.1. Data sheets are plenty of entities to be extracted and the majority of the relationship types between entities are clearly defined. The most common relationships between entities in data-sheets are "is-a", "is-part-of", "is-attribute-of", "is-capability-of" or its inverses such as "is-a-generalization", "has-part", "has-attribute" and so on. From now on, we will refer that type of documents as *data-rich* documents.

Almost all mentioned entities and relations in data-rich documents are able to be extracted and practically the entire document contains target entities to be extracted. Due to this fact, IE processes in data-rich documents might have better coverage and understanding of documents. Consequently, it is possible to think that IE over data-rich documents is not a machine-reading process as

Figure 1.1: Samples of data-rich documents (laptop data-sheets).

a)

Satellite A205 Series Detailed Product Specification

Model Name:A205-S4577	**Part Number:** PSAF0U-01Q009	**UPC:** 032017833104

Operating System C1 2
- Genuine Windows Vista™ Home Premium(32-bit version)

Processor and Chipset3

Intel® Centrino® Duo Mobile Technology featuring:
- Intel® Core™ 2 Duo Processor T5300[3]
  - 1.73GHz, 2MB L2, 533MHz FSB with 64-bit C1
- Mobile Intel® 945GM Express Chipset
- Integrated Wi-Fi® compliant wireless LAN
  Intel® PRO/Wireless 3945ABG (802.11a/b/g)

Memory[4]
- Configured with 1024MB PC2-5300 DDR2 SDRAM (both memory slots may be occupied). Maximum capacity 4096MB

Hard Disk Drive[5]
- 160GB (5400 RPM); Serial ATA hard disk drive; 9.5mm height; user removable

Fixed Optical Disk Drive[6]
- DVD SuperMulti (+/-R double layer) drive supporting 11 formats
  - Maximum speed and compatibility: CD-ROM (24x), CD-R (24x), CD-RW (10x), DVD-ROM (8x), DVD-R (Single Layer, (8x)), DVD-R (Double Layer, (4x)), DVD-RW (4x), DVD+R (Single Layer, (8x)), DVD+R (Double Layer, (4x)), DVD+RW (4x), DVD-RAM (5x)
- Data
  - i.LINK(tm) IEEE-1394
  - USB v2.0 – 4 ports

Physical Description
- Dimensions (WxDxH Front/H Rear): 14.3"x10.5"x1.32"/1.55"
- Weight: Starting at 6.29lbs depending upon configuration[11]
- LCD Cover Color: Onyx Blue Metallic

Power
- 65W (19V 3.42A) 100-240V AC Adapter.
  - Dimensions (WxDxH): 4.3" x 1.8" x 1.3"

Battery[12]
- 4000mAh Lithium Ion battery pack
  - Dimensions (WxDxH): 8.1" x .83" x 2.1"
  - Weight: starting at 0.68 lbs

Software C1 13
- Toshiba Software and Utilities
  - TOSHIBA Value Added Package
  - Electronic User's Guide
  - Bluetooth Stack for Windows by Toshiba
  - TOSHIBA ConfigFree®
  - TOSHIBA Assist
  - TOSHIBA Security Assist
  - TOSHIBA Disc Creator
  - TOSHIBA Extended Tiles for Windows Mobility Center

b)

HP Compaq 6510b Notebook PC

Operating system	Preinstalled: Genuine Windows Vista Business 32/64[1] Genuine Windows Vista Home Basic[1] Genuine Windows XP Professional FreeDOS	Supported: Genuine Windows Vista Enterprise Certified: SuSe Linux Enterprise Desktop 10
Processor[2]	Intel® Core™2 Duo (up to 2.4-GHz, up to 4-MB L2 cache), Intel® Core™2 Duo (up to 1.8-GHz, up to 2-MB L2 cache)	
Chipset	Mobile Intel® GM965	
Memory	DDR2 SDRAM, 667-MHz, two SODIMM slots supporting dual channel memory, 512/1024/2048/4096-MB	
Hard drive(s)[7]	SATA 80/120/160-GB 5400 rpm, 80-GB 7200 rpm, HP 3D DriveGuard	
Removable media[3]	12.7-mm optical drives: DVD+/–RW SuperMulti DL LightScribe, DVD+/–RW SuperMulti DL, DVD/CD-RW Combo, DVD-ROM	
Display	14.1-inch diagonal WXGA (1280 x 800 and 16M colors) 14.1-inch diagonal WXGA BrightView (1280 x 800 and 16M colors)	
Graphics[8]	Intel GMA X3100, up to 384-MB shared system memory	

shallow as IE over natural language text documents.

Data-rich documents in the *Information Technology* (IT) domain are frequently used to describe products with technical and commercial purposes. Those commercial data-sheets are relevant in e-commerce environments because of people decides if some product is the one that they are looking for based in the information contained in them . For instance, an buyer in the Internet looking for a laptop using a price engine such as Pricegrabber [1] can find hundreds of options with tens of features for each product data-sheet.

To read data-sheets in an automatic way (i.e. extract information) is the first step to assist decision making processes. However, aided decision making is not the only use scenario; technological monitoring, information retrieval refinement, automatic document translation, question answering, information integration from texts, among others are applications for the information extracted from data-rich documents.

The problem that is addressed in this thesis is to extract all the product characteristics mentioned in a set of commercial data-sheets circumscribed to a sub-domain of the IT domain. The selected the laptop computers sub domain due the following reasons:

- Laptops are among the most popular products in the IT domain. Therefore, a wide range of products are available and the extracted information has special interest.
- Data-sheets that describe laptops have a considerable number of composing parts, physical characteristics and technical features. Thus, the methods, prototypes and conclusions developed in this thesis could be applied to products of similar or smaller complexity.

The problem of IE in documents has been addressed in the past using different techniques. Some approaches use machine learning models trained with a labeled corpus (see [17] for a survey). The limited availability of labeled corpus is generally a drawback and the process of building it for specific applications is slow and expensive. However, those methods can be applied in several domains and languages only with small changes. Another approach is to perform IE based on a set of fixed extraction rules and dictionaries or gazetteer lists [27]. Rule-based approaches are relatively easy to setup but experts with linguistics and specific domain skills are required for writing the rules, which also is, in fact, an expensive resource. Another approach is to seed a small labeled corpus [24] or set of rules and to use a bootstrapping strategy based on a large corpus or the web [35, 23, 13]. This is an effective way to address the drawbacks of previous approaches. Nevertheless, the size of the corpus has to be considerable in order to achieve enough statistical evidence in order to make good generalizations. In addition, the bootstrapping approaches based on the web require a huge quantity of queries to search engines that commonly restrict the number of queries.

[1] `http://computers.pricegrabber.com`

In general, the previously mentioned IE approaches exploit context, order in text sequences and dictionaries in order to extract the target information in poorly structured text or free text. There are another IE approach based on the structural homogeneity of the documents. This IE systems are known as *wrappers* [49, 20]. Wrappers exploit the visual formatting regularity of the web pages in order to identify extraction targets.

The data-sheets documents that are being considered in this thesis have a particular set of characteristics and challenges:

- The formatting homogeneity is not warranted. Even the documents generated by the same brand for similar products have several differences in structure, style and appearance.
- The number of extraction targets is high. A typical laptop data-sheet can have more than a hundred of extraction targets.
- The density of the extraction targets is high. The documents are plenty of product features with long sequences of consecutive targets. Great part of the IE approaches builds separate extraction models for each individual target [37, 36]. This approach does not seem to be convenient in data-sheets because the target density makes some targets become context the others.
- The ambiguity of the targets is high. The ambiguity in target happens when targets of the same time has different meanings. For instance, extracting the targets *date-of-the-news* and *date-of-the-event* from a news corpus, the dates have to be identified and further disambiguated. The measures in laptop data-sheets have the same type of ambiguity. For instance, memory measures (e.g. 512MB) can stand for installed memory, cache memory, size of the installed memory modules, memory requirements of a particular included software, video memory, etc. The same type of ambiguity happens in other measures such as distances, weights, voltages and more.
- High number of targets plus high density plus ambiguity means high labeling cost. To provide a labeled corpus for model construction or evaluation is particularly an expensive task.
- The use of acronyms, abbreviations morphological variations in general are extensive in data-sheets. For instance, "MS Win XP Pro Ed." stands for "Microsoft Windows XP Professional Edition".
- The availability of large quantity of data-sheets for different sub-domains is not warranted. Whereas, it is possible to gather some hundreds of laptop data-sheets in English, only a few tens can be found for blade-servers.

In order to face those challenges, a new approach that poses the IE task as a *word sense disambiguation* (WSD) problem is proposed in this thesis. WSD

[60] is a natural language processing (NLP) task that aims to select the correct sense combination for all polysemous words in a text. We noticed the following similarities between WSD and our particular IE problem:

- The majority of the verbs, nouns and adjectives in a text are polysemous. Each polysemous word is a disambiguation target that has to be labeled.
- The density of polysemous words in a text is high.
- The number of senses for each polysemous word can be very high if the word senses are finely defined.
- The selected sense combination reflects the semantical coherence of the text sequence. Similarly, the product features in a data-sheet have a coherence sequence clearly related with their meaning.
- Verbs, names and adjectives are subject to morphological variations such as plurals, conjugations and gender.

A particular set of approaches that address WSD use as disambiguation resource a general lexicalized ontology called WordNet [62]. WordNet is a semantic network whose nodes are concepts and edges are semantic relationships such as hypernymy ("is-a-generalization-of"), hyponymy (i.e. "is-a"), meronymy ("is-part-of"), holonymy (i.e. the opposite of meronymy), synonymy and antonymy. Besides, each concept has a set of lexicalizations in a particular language such as English.

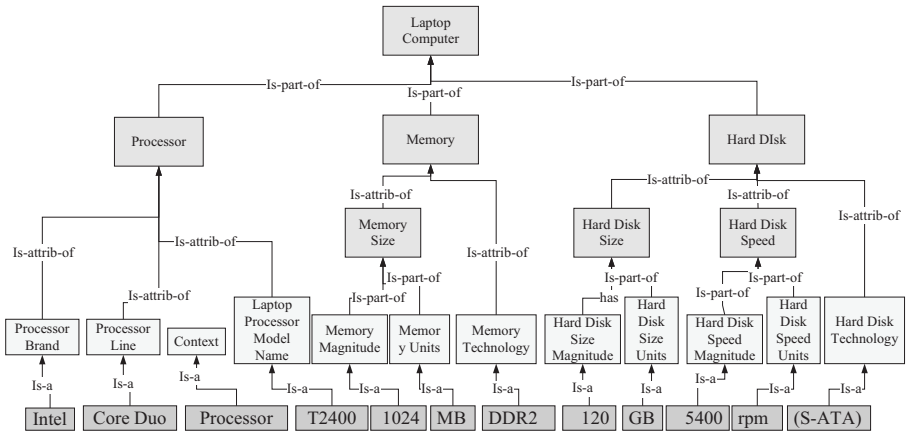
The ontologies can also be used to model concepts in specific domains and their use have become popular with the arrival of the semantic Web [11]. Ontologies have been used in the past for IE with different roles: as repository of the extraction model [33], as semantic dictionary of instances [74, 31, 57], as schema to be populated with instances [42, 88], as source of inferences in extracted information [16]. Only a few approaches have used ontologies for guiding the extraction process exploiting the ontology graph information [79, 96].

As part of the proposed solution, a laptop ontology was built by the authors from scratch, describing thoroughly all the attributes and composing parts of a laptop computer. Next, it was lexicalized with terms in English according to a set of data-sheets gathered from Internet. This lexicalized ontology is used as input to an information extraction system that labels the extraction targets using a WSD-inspired algorithm. On the other hand, the morphological variations associated with the acronyms and abbreviations were addressed using a strategy based in approximate string matching. The final output of the proposed system is an unambiguous labeling over the text of the data-sheets. A sample of an unambiguous labeling example is shown in figure 1.2.

1.1 Research Questions and Contributions

Firstly, the approximate string matching problem in the name matching task were thoroughly reviewed in order to provide a strategy to deal with the morpho-

Figure 1.2: Example of an unambiguous labeling over a document sequence sample.



logical variations of the terms mentioned in data-sheets. Particularly, we were focused in the following question: Is it more convenient to treat text strings to be compared as sequences of characters or sequences of tokens (i.e. words) in the name matching task? In order to answer this question different string comparison methods at character- and token-level were studied. Particularly, a new general combination method based in fuzzy-cardinality for character and token level measures were proposed in this thesis. Experiments carried out with 12 name matching data sets well studied in the related literature showed that the proposed method consistently outperformed other methods or reached similar results. The contribution made by this new method to the string processing field has application in our specific term matching problem and additionally in fields such as natural language processing, bioinformatics and signal processing as well.

The approximate string matching bring us the ability to detect terms that have misspellings, OCR errors, typos and morphological variations. However, it is unavoidable that very similar strings could have no relation at all. Besides, given our specific information extraction task, the approximate string matching provides flexibility for detecting terms, but it also provides noise. Some of the experiments carried out in this work were designed to research about this trade-off.

Secondly, motivated by the specific problem that it is being addressing in this thesis, we explored the applicability of the WSD techniques to the term disambiguation problem in the context of an IE task. The main question is: Is the proposed specific information extraction problem analogous to the NLP's Word Sense Disambiguation problem? Our contribution to the IE field is to explore that question providing a new way to address the extraction problem when the number and type of targets is considerable.

Finally, another new ideas were proposed in this thesis but they were not

exhaustively compared with other similar approaches due the scope of this work.

- A new semantic annotation schema for documents based on *semantic paths*. This annotation schema allows unambiguous semantic labels with independence of the design of the ontology or the document format.
- The use of *shortest path* as disambiguation strategy in sequences of words/tokens or terms.
- The *acronym sensitive name matching* were proposed as a method to integrate acronym matching heuristics to any approximate string matching technique based on tokens.

1.2 Thesis Structure

This document is structured as follows.

The chapter 2 presents the developed laptop ontology, lexicon and evaluation document corpus. That resources are the input data for the proposed IE prototype. That chapter describes and quantifies the ontology, lexicon and the labeled document included in appendices A, B, and C.

The chapter 3 studies the name matching problem using static approximate string matching methods. This chapter can be read as a separated paper with only a few references to the rest of the document. The experimental evaluation of the proposed methods was made with data sets used in others previous studies by researchers in the fields of record linkage and name matching.

The chapter 4 the background and relevant previous works related to knowledge-based WSD is presented. In addition a new graph disambiguation method based on shortest path is proposed.

The proposed IE system is presented in chapter 5. This chapter assembles the building blocks presented in chapters 3 and 4 in a coherent system. Additionally, an experimental evaluation was made in order to assess the proposed methods for: approximate string matching, semantic relatedness measurement and disambiguation strategy. The results obtained using those methods were compared against baselines.

In chapter 6 the methods for acronym matching were reviewed and some of the heuristics presented for those approaches were used to propose a method to integrate the acronym matching to the name matching task. The proposed method were integrated to the system presented in chapter 5 and its performance where assessed comparing its results against the best results obtained in that chapter.

Finally, chapter 5 gives the conclusions of this thesis and and discusses briefly the future work.

Chapter 2

Domain Ontology, Lexicon and Corpus

The construction and preparation of the input data for the information extractor system proposed in this thesis is presented in this chapter. Firstly, some basic concepts related to ontologies in the sense of a formalism for representing knowledge are presented. Also we introduce some general issues about ontologies their constructors and formal definition in section 2.1 in order to make understandable the next sections.

The system presented in this thesis was built from scratch. Thus the ontology, lexicon and evaluation corpus were manually prepared by us. This process is briefly described in this chapter. In section 2.2 the *laptop ontology* is presented with the chosen formalism to represent it. Basically, the laptop ontology describes in detail how a laptop computer is composed of and its attributes and those of its composing parts. In section 2.3 the *laptop lexicon* is presented. This lexicon is a dictionary with the representation in English terms of the concepts included in the laptop ontology. Finally, in section 2.4 the selected data-sheet corpus is presented with the selected annotation schema.

2.1 Ontologies

Ontologies are artifacts that are useful to describe real word entities and domains. In computer science, ontologies are graphs in which their nodes (i.e. vertices) are entities or concepts, and edges correspond to semantic relations between them [41]. Ontology definition languages offer a wide set of constructors allowing complex ontology modeling. The most common type of relations used in ontologies are hypernymy (i.e. "is-a") and meronymy (i.e. "is-part-of" or its inverse "has-part"). The former allows to define an entity class hierarchy and the latter describes the properties of each concept. Other common ontology constructors are cardinality restrictions on meronymy relations (e.g. a laptop has to have only one "display panel" but can have zero to three "PCCard" slots). Other

important quality of the ontologies is its ability to obtain new knowledge using logical inference or reasoning. For that, many ontology definition languages include the use of logical axioms such as universal and existential quantifiers *i.e.* $\forall\exists$.

The knowledge included in domain ontologies describes objects and domains with concepts and relations. The names of those concepts and relations are only reference names assigned by the person who builds the ontology. For instance, the concept that represent a hard disk can be named like "HardDisk" of "concept_235". Clearly, good practices in ontology construction recommends that the names of the concepts being represented are related to their names in the real world. Although, the ontologies model real-word entities and concepts there is not compromise of how those entities are referred in an specific language such as English.

2.1.1 Ontology Constructors

The ontology constructors are the building blocks of the ontologies. Popular ontology definition languages such as OWL[1] have consolidated many of the most used ontology constructors. Some of those constructors are listed as follows:

Class Definition declares and names the definition of a set of entities that represents a class.

Class Hierarchy is the definition of a class taxonomy tree or directed acyclic graph (DAG) ranging from general classes to specific ones. Particularly, class hierarchies are built with "is-a" relations, but those relations are usually considered as a primitive constructor with special inheritance behavior.

Object Properties defines and names the relations between objects.

Domain/Range Restrictions defines the allowed classes in the left or right parts of a relationship.

Cardinality Restrictions restricts the possible number of relations of the same type that an entity can have.

Universal Restrictions are logical axioms (*i.e.* $\forall$) restricting a relation with a condition that has to be satisfied by all the instances of a class.

Existential Restrictions are logical axioms (*i.e.* $\exists$) restricting a relation with a condition that has to be satisfied by at least by one of the instances of a class.

Sets Operations are class definitions based in previously defined classes and set operations *e.g.* $\cup$, $\cap$.

[1] http:/www.w3.org/2004/OWL/

2.1.2 Formal Definitions

For the sake of clarity, a mathematical definition of the ontology model used in this thesis is presented in this section. We define an ontology $O = \{S, C, h, g\}$ where:

- S is the root concept (e.g. *laptop*).
- C_p is a set of internal or upper level concepts (e.g. *memory*, *hard_disk*, *display*, *megabyte*).
- C_t is a set of terminal or lower level concepts (e.g. *megabyte*) that are not "divided" by hypernymy, holonymy or attribute relationships.
- h is the hypernymy (i.e. is-a) function $h : (C_p \cup S) \rightarrow (C_p \cup C_t)$ that established a taxonomy of concepts and instances (e.g. *Computer*$\rightarrow_h$*Laptop*, *StorageDevice*$\rightarrow_h$ *HardDisk*, *MemoryUnit*$\rightarrow_h$ *MegaByte*).
- g is the holonymy-/attribute-like (i.e. has-part, has-attribute) function $g : C_p \rightarrow (C_p \cup C_t)$ that establishes a composition hierarchy (e.g. *Laptop*$\rightarrow_g$ *Display*, *Display*$\rightarrow_g$*DiagonalLength*).

The lexicalization for an ontology is the addition of a dictionary of terms mapping the concepts modeled in the ontology in a language such as English. We extend the previous ontology definition O to $O_L = \{S, C, h, g, L, f\}$ in order to include the lexicon information as follows:

- L is a set of lexicalizations of concepts names (e.g. lexicalizations related to the *Brand* concept are*"brand", "manufacturer", "builder", "made by"*; related to *HP* concept *"HP", "Hewlett Packard", "HP corp."*)
- f is the lexicalization function $f : (c_1, c_2, \ldots, c_n) \rightarrow L$ with $n \geq 1$ where $c_i \in C$ and $c_1, c_2, \ldots, c_n$ is a valid chain of concepts linked by h or g relations. This function allows simple lexicalizations such as (*HP*)$\rightarrow_f$ {*"Hewlett Packard"*, "*HP*"} or more informative lexicalizations such as (*Software*$\rightarrow_g$*Brand*$\rightarrow_h$*Microsoft*)$\rightarrow_f${*"MS", "Microsoft"*}.

The definition of O is constrained to avoid cycles in the h and g functions in agreement with real world modeling cases. The ontology O can also be represented as a graph where the nodes are the sets S, C and edges are h and g functions. This graph is also directed and without cycles, i.e. DAG (directed acyclic graph).

2.2 The Laptop Ontology

The laptop ontology is a fine grained description of all the parts and attributes of a laptop computer. The model went beyond the physical description including additional attributes such as prices, commercial issues, warranty terms, etc. The

aim of the ontology is to model all the concepts that are commonly included in the laptop data-sheets used with commercial purposes.

The first step in the construction methodology of the ontology was to collect 30 laptop data-sheets during the month of May 2007 from the web sites of three popular laptop brands (i.e. Hewlett Packard[2], Toshiba[3] and Sony[4]). Those documents were used to build from scratch the ontology trying to identify all the mayor concepts (i.e. processor, hard disk memory, etc.). Next, each mayor concept was decomposed reviewing all the descriptions in the documents. The ontology construction was not a linear process, but it had many feedback processes until an operative version was obtained. Additionally, the concepts included in the ontology were not limited to the concepts mentioned in the documents. For example, a complete system of units of memory, distance, time, weight were defined even though some of those units were not cited in the documents.

The built ontology can be represented as a directed acyclic graph. The figure 2.1 shows as illustration a small sample of that graph. Basically, the graph contains three types of nodes and two type of edges. There is a "root" node associated to the concept "Laptop" which has not incoming edges. The other two types are the internal nodes that has both incoming and outgoing edges, and the terminal nodes that have only incoming edges.

The edges in the ontology graph were mainly "has-part" relationships shown with gray arrows in figure 2.1. Actually, the "is-a" relationships (represented with black arrows) were not used to model a class hierarchy but to model some relations close to the terminal nodes. The reason to use the "is-a" relationships in that way was due to the fact that the candidate "is-a" hierarchies were too narrow and did not offer to much added value. On the other hand, the composition hierarchy made with the "has-part" and ""has-attribute" relationships was depth and carried important information. Additionally, a semantic network with only one type of relationships is more convenient for the design of semantic relatedness metrics between concepts (see section 4.1).

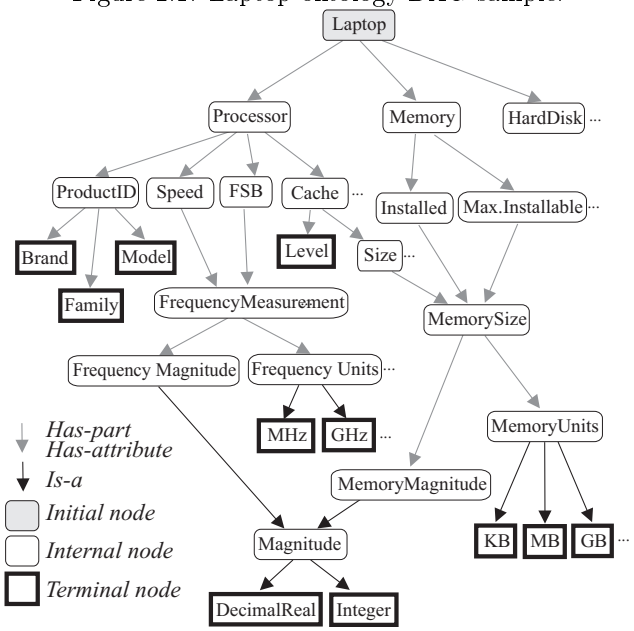
The laptop ontology is provided as reference in the Apendix A. The most common format to represent ontologies in text is using a set of ordered triples {$entity_1$, *relationship*, $entity_2$}. However significantly higher legibility is obtained with the format {$entity_a$, *relationship*, [$entity_1$, $entity_2$,...]}. This mean that the entity_a shares the same relationship with all the entities of the list. For instance, {*Laptop*, *has-part*, [*Processor*, *Memory*, *HardDisk*, ...]}. In addition, the constructions were listed in a top-down approach presenting first the constructions related to more general concepts.

Next some data related with the ontology graph are presented in order to figure out the general shape of the graph.

Number of nodes: 1 initial node, 237 internal nodes and 118 terminal nodes. Total number of nodes: 355.

[2]http://www.shopping.hp.com/notebooks
[3]http://www.toshibadirect.com/laptops
[4]http://www.sonystyle.com

Figure 2.1: Laptop ontology DAG sample.



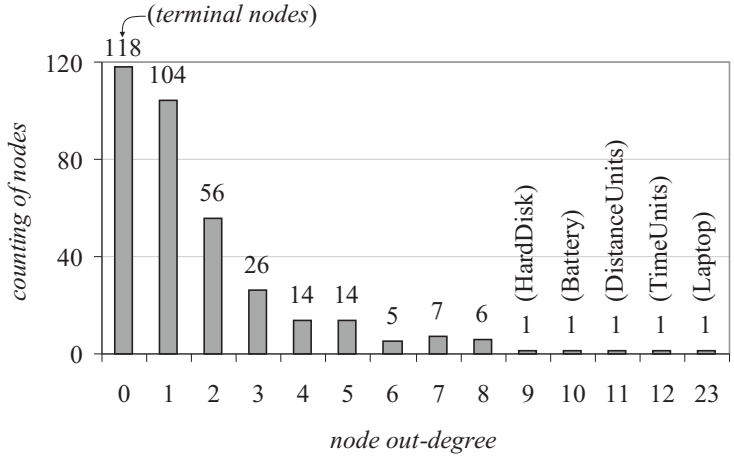
Number edges: 612

The figures 2.2 and 2.3 show charts of the *out-degree* and *in-degree* of the ontology graph nodes. The *out-degree* and *in-degree* of a node are respectively the number of outgoing and incoming edges of a node. In figure 2.2 is shown how the counting of nodes in each out-degree level is directly relate with the specificity of the node. For instance the more general node *Laptop* has the highest out-degree and the more specific nodes (i.e. terminal nodes) are "sinks" or nodes without outgoing edges.

Number of Semantic Paths: 1342 (i.e. number of different paths from the node *Laptop* to any terminal node).

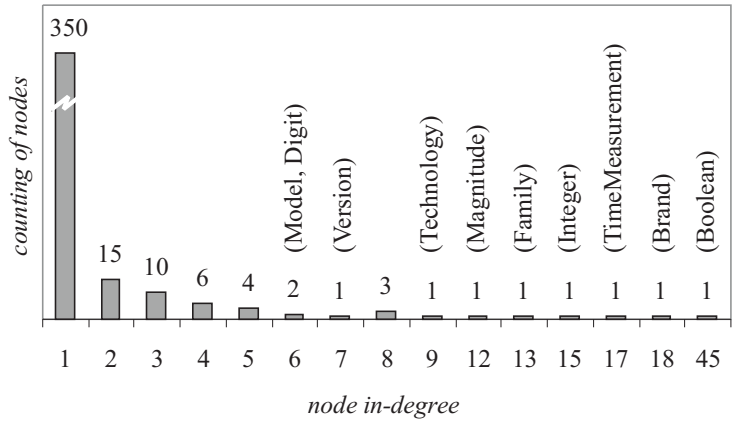
The *semantic paths* are unambiguous chains of concepts that link the root concept *Laptop* with a terminal node. The chart in figure 2.4 gives an idea of the depth of the graph. Another important issue related to the semantic paths is the number of semantic paths for each terminal node. This issue is important because of the more semantic paths are possible for a terminal node the higher its "use" ambiguity. The chart with the counting of terminal nodes grouped by their number of semantic paths is shown in figure 2.5. That chart shows that only 49 of the 118 terminal nodes are unambiguous. Thus, it is possible to say that the laptop ontology has an ambiguity of 58.47% at terminal nodes level, or ambiguity at terminal use.

Figure 2.2: Node counting chart for each out-degree level in the ontology graph.



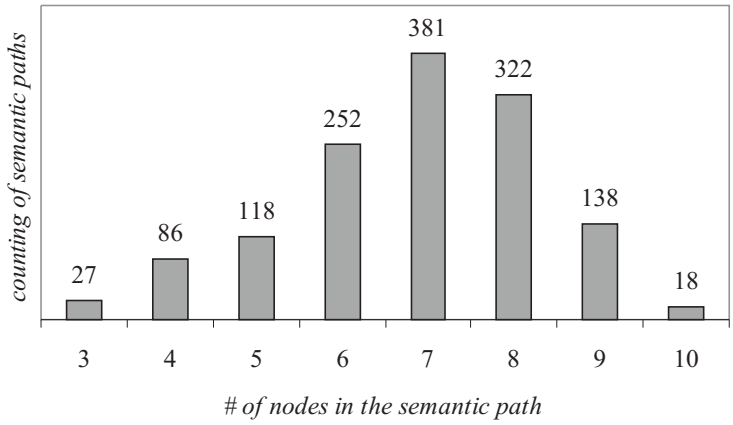
(e.g. there are 56 nodes with out-degree measure of 2)

Figure 2.3: Node counting chart for each in-degree level in the ontology graph.



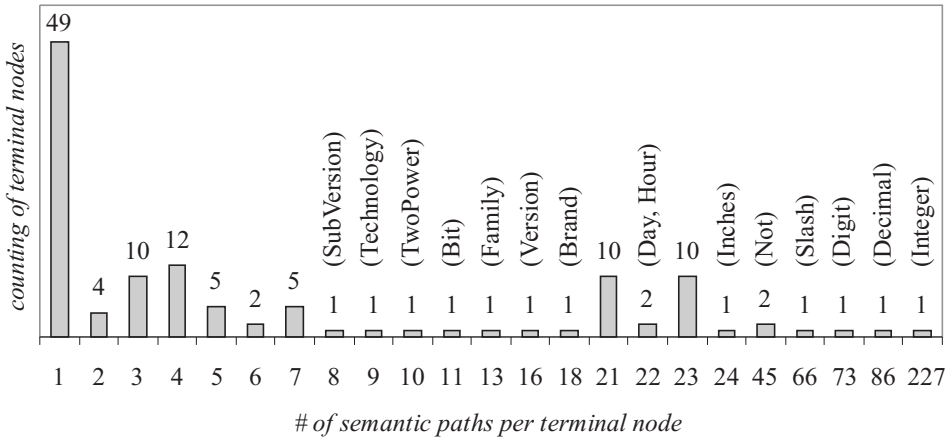
(e.g. there are 15 nodes in the ontology graph with in-degree measure of 2)

Figure 2.4: Semantic path counting chart for different semantic path lengths in the ontology graph.



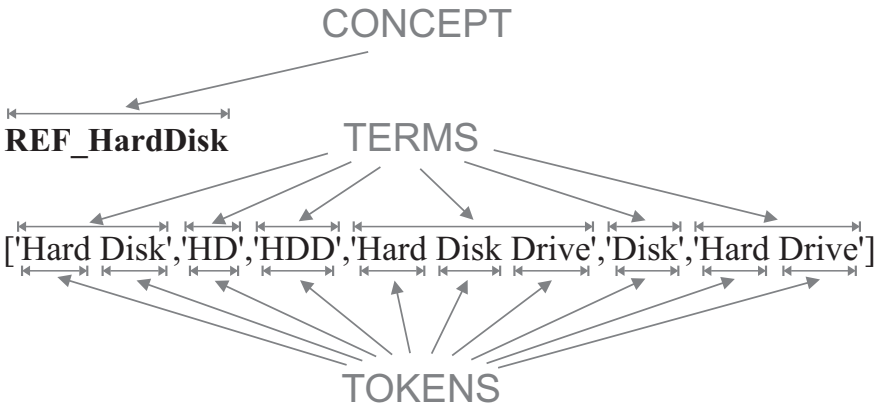
(e.g. there are 86 semantic paths with length of 4 nodes)

Figure 2.5: Terminal nodes counting chart for different number of their semantic paths.



(e.g. there are 10 terminal nodes in the ontology with three semantic paths)

Figure 2.6: Laptop lexicon entry example.



2.3 The Laptop Lexicon

The laptop lexicon was built simultaneously and with the same methodology used in the laptop ontology construction. Similarly to the laptop ontology the reach of the terms included in the lexicon went beyond the selected data-sheets particularly with the concepts related to measures and units.

The aim of the lexicon is to provide terms in English linked to concepts in the ontology. For instance the ontology concept HardDisk is related in English with the terms "Hard Disk", "Hard Drive" or with the acronyms "HD" or "HDD". Clearly, the terms can be composed of one or more words. We used the term *token* for representing words because in this thesis some special characters are considered "words". For instance, the term "Built-in" can be tokenized as {"Built", "-", "in"}.

Each lexicon entry is headed for a concept name and contains an unordered list of terms. The figure 2.6 depicts a lexicon entry with the roles of the concept, terms and tokens.

The laptop lexicon includes two types of lexicalizations one for *schema* and the other for *data*. The schema lexicalizations can be linked to any concept (node) in the ontology even to the root concept *Laptop* and the terminals. They were called schema lexicalizations because those terms refer to the names of the targets that we are interested in extract. The schema lexicalizations can be identified by the prefix "REF_" added to the concept name in the head of a lexicon entry. The lexicon entry shown in figure 2.6 is a schema lexicalization.

The data lexicalizations refer to the extraction targets such as numbers, magnitudes, units, brand names, technology names etc. The data lexicalizations are only linked to terminal nodes in the ontology. However, a terminal node can be referred by both schema and data lexicalizations. For instance, the terminal *Brand* has the schema lexicalization *REF_ Brand*:["Brand", "Made by", "Manufactured by"] and the data lexicalization *Brand*:["HP", "Sony", "Toshiba",

Table 2.1: Concepts lexicalized with regular expressions in the laptop lexicon.

Concept	Regular Expression
Integer	'^[0-9]+$'
Decimal	'^[0-9]+\.[0-9]+$'
UPCCode	'^[0-9][0-9]{10}[0-9]$'
Version	'^([0-9]+\.)+([0-9]+|x|a|b|c|d)$'

"Dell"].

As it was mentioned in section 2.1.2, more informative lexicon entries were allowed including in the head a list of concepts. For instance, the lexicalization *ProcessorID_Brand*:["Intel", "AMD"] refers only to processor brands. Also, schema lexicalizations are allowed of have head entries with multiple concept names like the following example: *REF_Modem_Version*: ["Version", "Speed", "Compliant"].

For some special concepts, it is not possible to enumerate all their possible related terms. For instance, the concept *Integer* clearly can not be enumerated. In order to address this problem, regular expressions were allowed as terms in lexicon entries. The concepts with terms expressed as regular expressions are listed in table 2.1.

The complete version of the laptop lexicon is presented in the Appendix B. In the same way that the ontology were outlined with some data, the following counting over the complete lexicon can help to figure out the structure of the lexicon.

Total number of lexicalized concepts: 383 (lexicon entries)

Total number of terms: 1700

Number of different terms: 1532

Total number of tokens: 2771

Number of different tokens: 1374

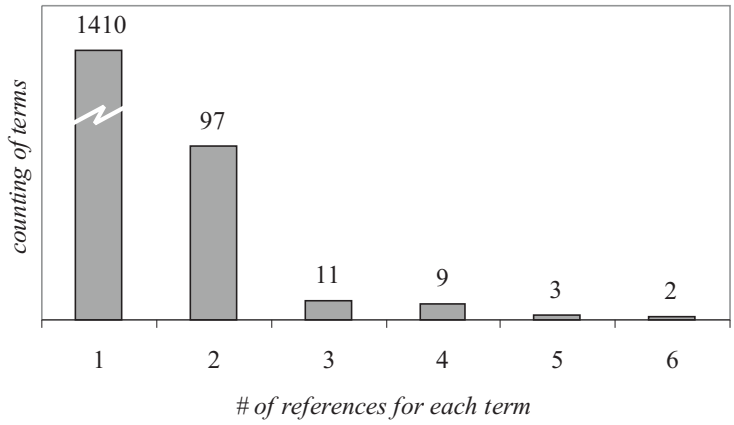
Number of one-letter acronyms: 70 (e.g. 'A' for the concept *Ampere*)

Number of acronyms: 480

It is worth to note that there were a considerable difference between the counting of terms and the counting of different terms. This mean, that many terms were included in more than one lexicon entry. The figure 2.7 depicts the number of terms according to the number of times that are referenced in lexicon entries. Only 1410 terms are unambiguous. Thus, the lexicon has an ambiguity of 17.06% at term level.

Another issue is the number of terms by lexicon entry illustrated in figure 2.8. That chart shows that only 49 concepts which have one term, the remaining concepts are lexicalized with several terms. Many of these lexicalizations are

Figure 2.7: Term counting chart grouped by the number of times that terms are referred in the laptop lexicon.



(e.g. there are 97 terms that have been referred in two lexicon entries)

Table 2.2: Document identification of the evaluation document corpus.

Doc ID.	Brand	Family	Model	Date (mdy)
1	Hewlett Packard	Pavilion	dv9500t	06/05/2007
2	Toshiba	Satellite	A130-ST1311	11/05/2007
3	Sony	VAIO	VGN-TXN15P/B	06/05/2007
4	Toshiba	Satellite	A135-S4498	11/05/2007
5	Hewlett Packard	Pavilion	dv9330us	06/05/2007

morphological variations of a same term. The most common of these variations are the basic term and its acronymed form.

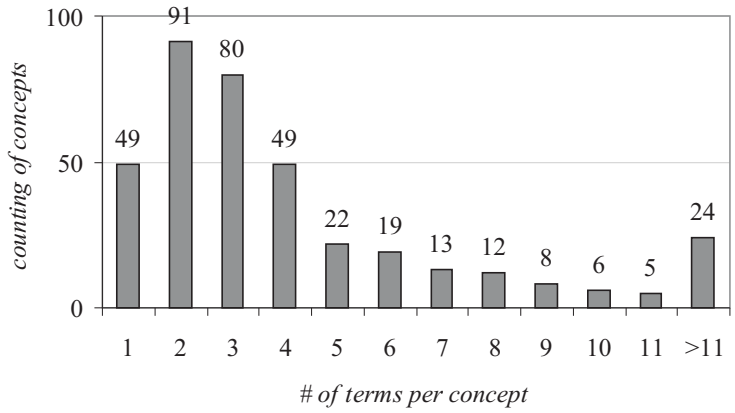
The number of token per term is depicted in figure 2.9. Almost half of the terms have more than one token. This fact motivated the content of chapter 3 where the multi-token approximate string comparison is discussed in depth.

2.4 Evaluation Document Corpus

In order to evaluate the information extraction system proposed in this thesis, it is necessary to provide a labeled set to compare with the output of the information extractor. Five data-sheets were chosen among the set of documents used to build the laptop ontology and lexicon to constitute the evaluation corpus. The laptop computers that those documents describe and the date of download are shown in table 2.2.

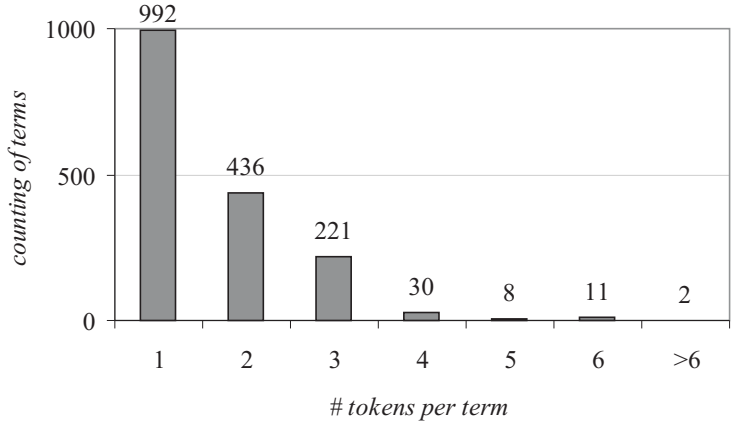
The documents were labeled so that all the relevant tokens had a label.

Figure 2.8: Concept counting chart grouped by the number of terms in each concept in the laptop lexicon.



(e.g. there are 91 concepts with 2 terms)

Figure 2.9: Chart of the counting of terms grouped by their number of tokens in the laptop lexicon.



(e.g. there are 436 terms with 2 tokens)

Table 2.3: Internal description of the evaluation document corpus.

Doc ID.	#tokens	#tokens target	#targets	schema targets	data targets
1	1925	744	536	31.90%	68.10%
2	744	363	246	43.09%	56.91%
3	921	462	313	47.28%	52.72%
4	801	416	267	47.19%	52.81%
5	740	327	234	38.89%	61.11%
TOTAL	5131	2312	1596	40.23%	59.77%

Relevant tokens are those that belong to some extraction target. In the same way that lexicon entries were divided in *schema* and *data* entries, extraction targets are also separated by the same criteria. In table 2.3 the total number of tokens, targets and the percentage of schema and data targets are reported for each document. The manually labeled document #1 is included as reference in Appendix C.

As it can be seen in the Appendix C, the text were extracted from the original documents so that only text, punctuation marks and *newline* characters were considered. All other formatting and structure tags were removed.

2.4.1 The Tokenizer

To tokenize a text means to divide the original character sequence into a sequence of sub-sequences called tokens. The text tokenizer was designed to fit the special needing of the IT data-sheets and it is based in the following static rules:

- One or more consecutive space characters are token separators.
- Any of the following characters are token separators and constitute themselves a one-character individual token: {'(', ')', '[', ']', '$', '%', '&', '+', '-', '/', ';', ':', '*', '?', '!', *coma* , <*cr*>, <*lf*>, <*tab*>, *quote*, *double quote*}.
- Consider period ('.') as token separator if it has at its right, left of both sides character spaces.
- Divide into two tokens a consecutive digit and an alphabetic character, e.g. "2GB" and "5400rpm" separates as "2", "GB" and "5400", "rpm" respectively.
- Divide into three tokens a 'x' character surrounded by digits, e.g. "1200x768" separates as "1200", "x", "768".

The data-sheet documents in the corpus were tokenized using the those rules. Similarly, when the terms in the lexicon were treated as sequences of tokens, the tokenizing process were made with the same rules set.

2.4.2 Semantic Paths

As it was briefly mentioned previously, the *semantic paths* are a sequence of nodes starting with the initial node (i.e. *Laptop*) and finishing in any terminal node. This sequence has to be a valid path in the ontology graph. Let's consider the semantic paths in the sample graph in figure 2.1 for the terminal concept *MB* (i.e. mega bytes):

1. *Laptop*→*Processor*→*Cache*→*Size*→*MemorySize*→*MemoryUnits*→*MB*
2. *Laptop*→*Memory*→*Installed*→*MemorySize*→*MemoryUnits*→*MB*
3. *Laptop*→*Memory*→*MaxInstallable*→*MemorySize*→*MemoryUnits*→*MB*

Each one of those semantic paths are the unambiguous possible uses for the concept *MB*. Now, if an acronymed term such as "M.b." is linked in the lexicon to the concept *MB*, then those semantic paths are the possible uses (or senses) for the term "M.b.". Let's consider another example with the semantic paths for the terminal *Integer* in the same graph:

1. *Laptop*→*Processor*→*Speed*→*FrequencyMeasurement*→ *FrequencyMagnitude*→*Magnitude*→*Integer*
2. *Laptop*→*Processor*→*FSB*→*FrequencyMeasurement*→ *FrequencyMagnitude*→*Magnitude*→*Integer*
3. *Laptop*→*Processor*→*Cache*→*Size*→*MemorySize*→ *MemoryMagnitude*→*Magnitude*→*Integer*
4. *Laptop*→*Memory*→*Installed*→*MemorySize*→ *MemoryMagnitude*→*Magnitude*→*Integer*
5. *Laptop*→*Memory*→*MaxInstallable*→*MemorySize*→ *MemoryMagnitude*→*Magnitude*→*Integer*

2.4.3 Annotation Schema

The proposed annotation schema for the data-sheet documents consist of linking sub-sequences of tokens in the document to a semantic path. Let's consider the string "Installed Memory: 512MB/2GB(max)". The tokenized version of this string is:

["Installed", "Memory", ":", "512", "MB", "/", "2", "GB," "(", "max", ")"].

The labeled token sequence is as follows:

["Installed", "memory"]: *Laptop*→*Memory*→*REF_ Installed*

[":"]: no-label

["512"]: *Laptop*→*Memory*→*Installed*→*MemorySize*→ *MemoryMagnitude*→*Magnitude*→*Integer*

["MB"]: *Laptop→Memory→Installed→MemorySize→MemoryUnits→MB*

["/"]: no-label

["2"]: *Laptop→Memory→MaxInstallable→MemorySize→MemoryUnits→GB*

["("]: no-label

["max"]: *Laptop→Memory→REF_MaxInstallable*

[")"]: no-label

Chapter 3

Static Fuzzy String Searching in Text

The term *fuzzy string searching* (FSS) is used to refer the set of techniques that compare strings allowing errors. FSS is based on similarity metrics that compare sequences at character- and token-level. The static string metrics are those that compare two strings with some algorithm that uses only the information contained in both strings. On the other hand, adaptive methods are those that use additional information from the corpus in supervised [80, 12] or unsupervised forms such as the *Soft-TD-IDF* measure proposed by Cohen et al. or the Jensen-Shannon distance [25]. Adaptive methods outperform other static metrics in certain matching problems but have as drawbacks the cost to train the model, the problem of getting labeled corpora and their scalability. This chapter is focused in some widely known static string metrics.

This chapter also presents different methods for combining static fuzzy string metrics at character- and token-level. Many popular fuzzy string metrics (e.g. edit-distance) use characters as comparison unit but do not take advantage of the natural division in words (tokens) of text sequences considering the separator character (i.e. space) as an ordinary character. On the other hand, token level metrics consider words as comparison unit but most of them compare tokens in a crisp way. In many NLP task, qualities of both character and token-level metrics are desirable. Consequently the combination of those types of metrics is an interesting topic. Only a few combination methods have been proposed in the related literature. We tested and compared them with a new general method to combine cardinality based coefficients (e.g. Jaccard, dice, Sorensen, cosine, etc.) with character-based (e.g. edit-distance, Jaro-distance, n-grams, etc.) string metrics.

3.1 Introduction

String similarity metrics have been used to compare sequences in domains such as the biomedical and the natural language processing (NLP). Probably the most known character-level metric is the edit-distance originally proposed by Levenshtein (1965) [53] consisting of counting the number of edition operations needed to transform one string into other. The three basic operations are: insertion, deletion and substitution. Several modifications to the original edit distance have been proposed varying cost schemes and adding edit operations such as transpositions, opening and extending gaps (see [32] for a recent survey).

Most of the character-based metrics consider the strings to be compared as character sequences. That approach is very affordable when the strings to be compared are single words having misspellings, typographical errors, OCR errors and even some morphological variations. However, due to the inherent nature of the human language, text sequences are separated into words (i.e. tokens). This property can also be exploited to compare text strings as sequences of tokens instead of characters. That approach deal successfully with problems such as comparing strings with tokens out of order and with different number of tokens, and it has been used in fields such as information retrieval [6], record linkage [63], acronym matching [86] among others.

Depending on the specific matching task some metrics perform better than others but there is not one technique that outperforms consistently the others [12, 21]. For instance the edit-distance, Jaro distance and Jaro-Winker distance [92] are suitable for name matching tasks but perform poorly for record linkage or acronym matching tasks.

Approaches for combining [19, 64] character- and token-level metrics have been proposed in order to preserve the character-level metric properties but considering the token information. Nevertheless, the problem has not been studied in detail and there is not a general method to combine the known metrics.

In this chapter a new general method for using cardinality based metrics such as *Jaccard*, *dice* and *cosine* coefficients in combination with character-level metrics. Additionally, we explored the use of the *Monge-Elkan* combination method extended with the generalized mean instead of the simple arithmetic mean. The proposed approaches open a wide set of options with the possible combinations of known techniques that can be combined to deal with different matching problems in NLP.

This chapter is organized as follows. Section **??** review some of the most relevant static character- and token-level string metrics. In Section 3.3, we reviewed the known combination methods. In Section 3.5, our proposed new cardinality-based combination method is presented. Results from some experiments carried out using 12 data-sets are presented in Section 3.6. Finally, some concluding remarks are given in Section 3.7.

Figure 3.1: *Edit distance* dynamic programing matrix example

		S	**U**	**N**	**D**	**A**	**Y**
	0	1	2	3	4	5	6
S	1	0	1	2	3	4	5
A	2	1	1	2	3	3	4
T	3	2	2	2	3	4	4
U	4	3	2	3	3	4	5
R	5	4	3	3	4	4	5
D	6	5	4	4	3	4	5
A	7	6	5	5	4	3	4
Y	8	7	6	6	5	4	3

3.2 Static Fuzzy String Measures

Among the most popular character-level similarity metrics are the well known edit-distance [53], longest common sub-sequence [68], Jaro-Winkler distance [92] and n-gram distance [87]. Those metrics are useful to deal with typos and misspellings but our special interest is its ability to deal with inconsistent abbreviations, variable acronym generation rules and typographical variations.

Unlike character-based similarity metrics, token-level metrics uses tokens (*i.e.* words) as comparison unit instead of characters. Moreover algorithms structure is analogous to many character-based algorithms, but instead of using character comparisons an additional similarity metric between tokens has to be supplied [19]. That inter-token metric can again be other character-based approximate string metric.

3.2.1 Character-Based Measures

3.2.1.1 Edit-Distance Family

The *edit distance* was originally proposed by Levenshtein [53], and consist in counting the number of edition operations needed to transform one sequence in other. The tree basic operations are: insertion, deletion, substitution (or replacement). The algorithm to compute the edit-distance [90] is a dynamic programming solution that stores in a matrix the counting of edit operations for all possible prefixes of both strings. This algorithm computes the edit-distance between two strings $s_m$and $s_n$of length m and n in a time complexity of $O(mn)$ and in a space complexity of $O(\min(m,n))$. In Algorithm 1 an implementation of the edit-distance is shown and in figure 3.1 an example of the solutions matrix is shown.

The edit-distance has been extended in the operations and in the cost associated to them. For instance, the *Damerau/Levenshtein-distance*[28] considers an additional edit operation: the transposition operation *i.e.* swapping of two contiguous elements in the sequence. The *Longest Common Sub-sequence* [69, 4]

Algorithm 1 Edit distance

```
EditDistance(string1, string2)
   //declare matrix of integers
      d[0..len(string1), 0..len(string2)]
   for i from 0 to len(string1)
      d[i, 0] = i
   for j from 1 to n
      d[0, j] = j
   for i from 1 to len(string1)
      for j from 1 to len(string2)
         if string1[i] = string2[j] then
            cost = 0
         else
            cost = 1
         d[i, j] = min(d[i-1, j] + 1, // deletion
                       d[i, j-1] + 1, // insertion
                       d[i-1, j-1] + cost) // substitution
return (d[m, n])
```

can be viewed as an edit-distance with an cost schema allowing only insertions and deletions at cost 1 (the number of impaired characters). *Hamming distance* [81] *i.e.* string matching with k mismatches, allows only substitutions at cost 1 counting the number of terms in the same positions (this approach is only suitable for strings with the same length). Needleman and Wunsch also proposed a differentiate cost G (*i.e.* gap cost) for insertion and deletion operations; this approach is also known as Sellers Algorithm.

Waterman *et al.* [65] proposed an edit-distance measure adding an alphabet distance function $d(c_1, c_2)$, allowing different costs to the edit operations depending of the characters to be compared. For instance the substitution cost for the characters '1' and 'l' might be lower than the substitution cost between 'e' and 'h'.

Gotoh [40] modified the distance proposed by Waterman *et al.* considering open-/extend-gap operations with variable costs (o or Smith-Waterman-distance). These new operations allow closer matches with truncated or abbreviated strings (*e.g.* comparing 'Michael S. Waterman' with 'M.S. Waterman'). In order to have that property the cost of extending a gap have to be smaller than the gap opening cost.

Ristad and Yianilos [80] were the first in propose an adaptive method to learn the optimal costs for an edit-distance for a particular matching problem in a supervised way. They proposed an edit-distance with three operations: *insertion, deletion* and *substitution*. Let a and b be characters in the alphabet Σ of the strings including ϵ, the empty character. The costs to be learned are a matrix c of size $|\Sigma| \times |\Sigma|$ with the substitution costs at character level. The insertion cost for a character a is $c_{a,\epsilon}$. Similarly the deletion and substitution

costs are $c_{\epsilon,a}$ and $c_{a,b}$ respectively. The optimal costs are estimated using a training set with an expectation maximization (EM) strategy. Bilenko and Mooney [12] proposed a similar approach to learn the optimal cost for an edit-distance including open and extend-gap operations.

3.2.1.2 Jaro and Jaro Winkler Distances

The Jaro distance between two strings of length m and n takes only $O(m+n)$ in space and time complexity. It considers the number of common characters c and the number of transpositions t using the following expression.

$$sim_{Jaro}(s_m, s_n) = \frac{1}{3}\left(\frac{c}{m} + \frac{c}{n} + \frac{c-t}{c}\right)$$

The common characters are considered only in a sliding window of size $max(m,n)/2$, additionally the common characters can not be shared and are assigned with a greedy strategy. The case when $c = 0$, sim_{Jaro} returned a value of 0 avoiding division by zero. In order to compute the transpositions t two lists with the common characters are ordered according with its occurrence in the strings being compared. The number of non coincident characters in common positions in both lists are the number of transpositions t.

The Jaro-Winkler distance [92] is an improvement of the Jaro distance considering the number of common characters at the beginning of the strings l using the following expression.

$$sim_{Jaro-Winkler} = sim_{Jaro} + \frac{l}{10}(1 - sim_{Jaro})$$

3.2.1.3 q-gram Distances

The q-grams [48] are sub-strings of length q with in a string of length greater than q. Depending on q the q-grams are called uni-grams, bi-grams (or dig-grams), tri-grams and so on. For instance the trigrams in 'laptop' are 'lap', 'apt', 'pto', 'top'. Commonly, $q-1$ pad characters are added as prefix and suffix in order to consider strings smaller than q and to differentiate q-grams at the begin and the end of the string. The padded trigrams of 'laptop' are the trigrams of the string '##laptop##' using '#' as padding character.

When two strings are being compared a resemblance coefficient such as Jaccard, dice or overlap coefficient (see subsection 3.2.2) is used to compute a final resemblance metric between them. The time complexity of computing and comparing the q-grams of two strings of length m and n is $O(m+n)$.

Additionally to normal q-grams some variations have been proposed. *Skip*-grams [46] are bi-grams with one skip character in the center. For instance the 1-*skip*-grams of 'laptop' are 'l*p', 'a*p', 'p*o', 't*p'. With the exception of the q-grams containing padding characters, q-grams do not carry information about the position of the q-gram in the string. Another variation called *positional*-q-grams adds the position in the string to each q-gram and considers only common q-grams with a different in their positions lower than a preestablished value. A comparative study by Christen [21] showed that in a name matching data-set the

performance of different q-grams measures can be ranked as follows: positional-1-grams (better), 2-grams, positional-2grams, skip-grams, 1-grams, 3-grams and positional 3-grams (worse).

3.2.1.4 Bag Distance

Bartlolini et al. [9] proposed an approximation to the edit distance between two strings $s_m$and $s_n$of length m and n respectively, that can be computed in linear time with the following expression.

$$d_{bag}(s_m, s_n) = \max\{|s_m - s_n|, |s_n - s_m|\}$$

The operator '-' of the difference has a bag semantics. For instance the difference between the bags a, a, a, b and a, a, b, c, c is a. The bag distance first "drops" common characters and then takes the maximum of the number of elements in the asymmetrical differences.

The Bag-distance works as an approximation to the edit-distance but in less time. All edit-distance family measures are computed in $O(mn)$ and the bag-distance is computed in $O(m+n)$. Bag-distance has generally lower performance than edit-distance, but gives an efficient strategy to filter candidates to the same edit-distance and to other approximate string measures.

3.2.1.5 Compression Distance

Cilibrasi and Vitányi [22] proposed the idea of using data compression to obtain a distance measure between two objects x, y using the Normalized Compression Distance (NCD) computed with the following expression.

$$NCD(x, y) = \frac{C(xy) - min\{C(x), C(y)\}}{max\{C(x), C(y)\}}$$

$C(x)$ is the size of the compressed information of x, and $C(xy)$ is the same function but with the concatenation of x and y. The compressor C has to satisfy the conditions of a *normal* compressor:

1. *Idempotency.* $C(xx) = C(x)$, and $C(\lambda) = 0$, where $\lambda$is the empty string.
2. *Monotonicity.* $C(xy) \geq C(x)$.
3. *Symmetry.* $C(xy) = C(yx)$.
4. *Distributivity.* $C(xy) + C(z) \leq C(xz) + C(yz)$.

The data-compressor C for text strings and documents can be *BZ2*, *ZLib* or any other data-compression method. This method was originally tested by the authors in document clustering tasks. Christen [21] tested the method in a comparative study of approximate string metrics, obtaining comparable results against the best results obtained with measures of the Jaro and edit-distance families.

3.2.2 Token-Level Cardinality-Based Resemblance Coefficients

The token level measures consider the strings as sequences of tokens instead of consider them as sequences of characters. The most common approach is to handle the token sequences as sets and use resemblance coefficients to compare them.

The resemblance coefficients are quotients that compares two sets putting in the dividend a measure of comparison between them, and in the divisor a normalizing magnitude that represents both sets. In order to compute those measures the set operations *union*, *intersection*, *difference* and *complement* are used. Additionally to the set operations, the common $max()$ and $min()$ functions are used too.

Let A and B two multi-token strings. The set theory operations has particular meaning in set of tokens.

1. $|A|$ =number of different tokens in the A string.
2. $|B|$ =number of different tokens in the B string.
3. $|A \cap B|$ =number of different tokens in common between A and B strings.
4. $|A \cup B|$ =number of different tokens in common between A and B strings.
5. $|A \setminus B|$ =number of different tokens present in A but not in B.
6. $|A \triangle B|$ =number of different not common tokens in A and B.
7. $|A|^c$= number of different tokens that are not in the A string. Nevertheless, this operation is not practical in the task of compare relatively shot strings because the complement operation is made against a finite universe with cardinality n. This finite universe could be the entire data-set vocabulary that can be sin some cases a number considerably greater than the number of tokens of a specific set of tokens.
8. n =number of different tokens of the data-set.

Using the elements previously enumerated it is possible to build a wide range of resemblance coefficients. In table ,a list with some of the most known coefficients. The *name* column in that table contains the common name used by the scientific community or the reference name given by De Baets and De Meyer [5].

3.3 Combining Character-Level Measures at Token-Level

3.3.1 Monge-Elkan method

Monge and Elkan [64] proposed a simple but effective combination method to combine two token sequences. Given two sequences a, b and an external inter-

Table 3.1: Some expressions of resemblance coefficients.

#	Name	Expression	#	Name	Expression
1	*Jaccard*	$\frac{\lvert A\cap B\rvert}{\lvert A\cup B\rvert}$	15	R_9	$\frac{\max(\lvert A\backslash B\rvert^c,\lvert B\backslash A\rvert^c)}{n}$
2	*Dice*	$\frac{2\times\lvert A\cap B\rvert}{\lvert A\rvert+\lvert B\rvert}$	16	R_{10}	$\frac{\min(\lvert A\backslash B,\lvert B\backslash A\rvert)}{\max(\lvert A\backslash B\rvert,\lvert B\backslash A\rvert)}$
3	*cosine*	$\frac{\lvert A\cap B\rvert}{\sqrt{\lvert A\rvert\times\lvert B\rvert}}$	17	S_{17}	$\frac{\lvert A\cap B\rvert}{\max(\lvert A\backslash B\rvert^c,\lvert B\backslash A\rvert^c)}$
4	*Sorensen*	$\frac{2\times\lvert A\cap B\rvert}{\lvert A\cup B\rvert+\lvert A\cup B\rvert}$	18	R_{12}	$\frac{\min(\lvert A\backslash B,\lvert^c B\backslash A\rvert^c)}{\max(\lvert A\backslash B\rvert^c,\lvert B\backslash A\rvert^c)}$
5	NCD^1	$\frac{\lvert A\cup B\rvert-\min(\lvert A\rvert,\lvert B\rvert)}{\max(\lvert A\rvert,\lvert B\rvert)}$	19	R_{13}	$\frac{\min(\lvert A\backslash B,\lvert B\backslash A\rvert)}{\lvert A\triangle B\rvert}$
6	*overlap*	$\frac{\lvert A\cap B\rvert}{\min(\lvert A\rvert,\lvert B\rvert)}$	20	R_{14}	$\frac{\min(\lvert A\rvert,\lvert B\rvert)}{\lvert A\cup B\rvert}$
7	*Hamming*	$\lvert A\triangle B\rvert$	21	R_{15}	$\frac{\min(\lvert A\backslash B\rvert^c,\lvert B\backslash A\rvert^c)}{n}$
8	*matching*	$\frac{\lvert A\triangle B\rvert^C}{n}$	22	R_3^c	$\frac{\lvert A\cup B\rvert^c}{\min(\lvert A\rvert^c,\lvert B\rvert^c)}$
9	*Russell-Rao*	$\frac{\lvert A\cap B\rvert}{n}$	23	R_5^c	$\frac{\lvert A\cup B\rvert^c}{\lvert A\cap B\rvert^c}$
10	R_1	$\frac{\lvert A\cap B\rvert}{\max(\lvert A\rvert,\lvert B\rvert)}$	24	R_8^c	$\frac{\max(\lvert A\rvert^c,\lvert B\rvert^c)}{\lvert A\cap B\rvert^c}$
11	R_2	$\frac{\lvert A\triangle B\rvert^c}{\max(\lvert A\backslash B\rvert^c,\lvert B\backslash A\rvert^c)}$	25	R_{14}^c	$\frac{\min(\lvert A\rvert^c,\lvert B\rvert^c)}{\lvert A\cap B\rvert^c}$
12	R_4	$\frac{\lvert A\triangle B\rvert^c}{\min(\lvert A\backslash B\rvert^c,\lvert B\backslash A\rvert^c)}$	26	S_{17}^c	$\frac{\lvert A\cup B\rvert^c}{\max(\lvert A\backslash B\rvert^c,\lvert B\backslash A\rvert^c)}$
13	R_7	$\frac{\max(\lvert A\backslash B\rvert,\lvert B\backslash A\rvert)}{\lvert A\triangle B\rvert}$	27	S_{18}^c	$\frac{\lvert A\cup B\rvert^c}{n}$
14	R_8	$\frac{\max(\lvert A\rvert,\lvert B\rvert)}{\lvert A\cup B\rvert}$			

token similarity metric *sim*. The Monge-Elkan measure is computed as follows.

$$sim_{MongeElkan}(a,b) = \frac{1}{|a|}\sum_{i=1}^{|a|}\max{}_{j=1}^{|b|} sim(a_i,b_j) \tag{3.1}$$

In fact the Monge-Elkan measure is an approximation in time complexity of $O(mn)$ of the optimal assignment problem in combinatorial optimization. This approximation is reasonable due to the fact that the better solutions to the assignment problem would have a time complexity of $O(min(m,n)^3)$.

3.3.2 Token Order Heuristics

Christen [21] proposed two combination heuristic for character-based string metrics that are sensitive to the order of the characters such as Jaro and edit-distance family measures. Firstly, in order to compensate the weakness of those measures the strings to be compared are tokenized (separated in words) then ordered and joined again (*sorted-tokens*). The second heuristic consist of to list all token permutations for one of the string and compare the resulting strings with the second one (*permuted-tokens*). Finally the measure with the maximum value is chosen.

Two comparative studies [73, 21] showed that the permuting heuristic outperforms sorting. Unfortunately the complexity of exploring all order permuta-

tions is factorial, clearly un-practical when the number of tokens in the strings is considerable.

3.3.3 Token Edit-Distance

Chaudhuri et al. [19] proposed the idea of use the edit-distance with tokens as comparison unit instead of characters in a database data cleaning task. Analogously to the edit-distance, the three basic operations are *token replacement, token insertion* and *token deletion*. The costs of those edit operations are functions of tokens in question. For instance, the replacement cost can be a character based distance measure between the token to be replaced and the replacement token.

Chaudhuri et al. proposed costs for the insertion and deletion token operation weighted with the IDF (inverse document frequency) [45] factor, motivated by the idea that more informative tokens are more expensive to be inserted or deleted. However an static approach can consider insertion and deletion costs as constants.

3.3.4 Minkowski Family Metrics Combining character/token-level similarities

Vector space representation aims to express the strings to be compared as a pair of vectors with the same norm (i.e. number of elements). When it is being compared two sequence of tokens of the same size each token can be considered as a vector component. In the string matching problems considered in this chapter the vector space representation is not practical due to the fact that in the majority of the cases the number of tokens of the strings being compared are different.

However, that vector representation is useful in tasks such as record linkage and dictionary matching in documents. In the former task (record linkage), each field of a record is a vector component and usually is a multi-token string. In the later, each dictionary entry (pattern) has a fixed number of tokens usually much more small than the tokens in the document. Therefore, when the pattern is compared along the document sequence there are commonly enough tokens in the document to provide a vector with the same norm of the pattern. Although this combination method can not be applied in the context of this chapter, it is worth to review it because of the information extraction system proposed in this thesis has a dictionary (the laptop lexicon) to be matched with a set of documents.

Let two records with the same fields set $R_1 = \{a_1, a_2, \ldots, a_n\}$, and $R_2 = \{b_1, b_2, \ldots, b_n\}$; given a similarity function for each field i, $sim_i(a_i, b_i)$. This function $sim_i(a_i, b_i)$ can be any similarity measure of those presented in chapter . The problem is to obtain a total similarity measure $sim(R_1, R_2)$ between both records. The most common approach is to choose a metric from the Minkowski family of distance metrics defined by:

Table 3.2: Meaning of the Minkowski distance and generalized mean exponent.

p	meaning
$\rightarrow \infty$	maximum
2	quadratic mean
1	arithmetic mean
$\rightarrow 0$	geometric mean
-1	harmonic mean
$\rightarrow -\infty$	minimum

$$sim(R_1, R_2) = \left(\sum_{i=1}^{n} (sim_i(a_i, b_i))^p \right)^{\frac{1}{p}} \tag{3.2}$$

The Euclidean and City-block distances are special cases of the equation 3.2 when the exponent p is equal to 2 and 1 respectively. Dey and Sarkar [29] used the simplest measure from the Minkowski family with $p = 1$, but weighing the fields with the following weighed average $sim(R_1, R_2) = \sum_{i=1}^{n} w_i sim_i(a_i, b_i)$, having $\sum_{i=1}^{n} w_i = 1$.Weights $w_i$denote how much informative a field is.

The Minkowski distance is closely related with the concept of generalized mean. In fact, the only difference between the generalized mean expression (equation 3.3) and that of the Minkowski distance (equation 3.2) is the factor $1/n$. This factor preserves the output normalized if the values to be averaged were normalized.

$$\bar{x}(p) = \left(\frac{1}{n} \sum_{i=1}^{n} x_i^p \right)^{\frac{1}{p}} \tag{3.3}$$

The variable n means the number of values to be averaged and p is a parameter. For instance, when $p = 2$ the distances are combined with the euclidean distance; for a complete description of p see table 3.2.

3.4 An Extension to Monge-Elkan Combination Method

The Monge-Elkan method computes an arithmetical average of a sub-optimal assignment between the tokens of two strings given a inter-token similarity measure (see equation 3.1). The arithmetical mean gives a final measure where higher values are compensated with lower values of the similarity measure. However, this approach makes the assumption that higher inter-token similarities are as informative than the lower ones. Consider the following example:

$a =$"Lenovo inc."

$b =$"Lenovo corp."

$sim_{edit-distance}(a_1, b_1) = 1 - \frac{0}{6} = 1$

$sim_{edit-distance}(a_1, b_2) = 1 - \frac{5}{6} = 0.1666$
$sim_{edit-distance}(a_2, b_1) = 1 - \frac{5}{6} = 0.1666$
$sim_{edit-distance}(a_2, b_2) = 1 - \frac{4}{4} = 0$
$sim_{MongeElkan}(a, b) = \frac{1}{2}(\max(sim(a_1, b_1), sim(a_1, b_2)) + \max(sim(a_2, b_1), sim(a_2, b_2)))$
$sim_{MongeElkan}(a, b) = \frac{1}{2}(1 + 0.1666) = 0.5833$

However, the final measure of 0.5833 seems very low having in account that $sim(a_1, b_1)$is equal to 1. Additionally the simple edit distance measure looks more suitable in that case.

$sim_{edit-distance}$("Lenovo inc.","Lenovo corp.") $= 1 - \frac{4}{12} = 0.6666$

An approach that tries to promote higher similarities in the average calculated in the equation 3.1 can be the use of the generalized mean instead of the arithmetic mean. When the generalized mean exponent m is equal to 1 means the arithmetic mean, when $m = 2$ means the harmonic mean. In general, the higher m, the stronger the promotion of higher values in the mean. The generalized mean is computed with the following equation.

$$\bar{x}(m) = \left(\frac{1}{n} \cdot \sum x_i^m\right)^{\frac{1}{m}}$$

Lets consider $m = 2$ in the previous example:

$$sim_{MongeElkan2}(a, b) = \left(\frac{1}{2}\left(1^2 + 0.1666^2\right)\right)^{\frac{1}{2}} = 0.7168$$

This harmonic mean gives more importance to the number 1 than to 0.1666. We propose the idea that values of m greater than 1 can improve the performance of the matching. The expression to calculate the Monge-Elkan measure with the proposed extension can be written as follows.

$$sim_{MongeElkan_m}(a, b) = \left(\frac{1}{|a|}\sum_{i=1}^{|a|}\left(\max_{j=1}^{|b|} sim(a_i, b_j)\right)^m\right)^{\frac{1}{m}} \tag{3.4}$$

3.5 A New Cardinality-Based Token-/Character-level Combination Method

The resemblance coefficients are similarity measures used to compare sets. The Jaccard coefficient is one of them and is calculated with the following formula.

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

In order to compute the resemblance coefficients it is necessary a cardinality function |.| and only one basic set operation such as union or intersection due

to the set theory property: $|A \cap B| = |A| + |B| - |A \cup B|$. Using this property, the Jaccard coefficient can be written as follows.

$$Jaccard(A, B) = \frac{|A| + |B| - |A \cup B|}{|A \cup B|} \tag{3.5}$$

As it was explained in sub section 3.2.2 if a string is considered as an unordered set of tokens the resemblance coefficients can be used to compare multi-token strings. For instance *Jaccard*("Sergio Jimenez", "Sergio Gonzalo Jimenez")= $\frac{2}{3} = 0.6666$, that looks quite reasonable. However, if my name is written with two common misspellings in Spanish *Jaccard*("Cergio Gimenez", "Sergio Gonzalo Jimenez")= $\frac{0}{5} = 0$, this expected result is due to the fact that the identity function used in classic set operations is a crisp function. Using the concept of identity of the classic set theory it is clear that |{"Jimenez"}|=1, |{"Gimenez"}|=1, |{"Jimenez Jimenez"}|=1 and |{"Jimenez Gimenez"}|=2.

Now, lets consider two multi-token strings A and B with m and n tokens respectively. The set of token of A is $a_1, a_2, ..., a_n$and $b_1, b_2, ..., b_m$for B. The set of tokens of the concatenation of both strings $A \oplus B$ is $A \cup B$. Clearly, the concatenation operator $\oplus$takes care of insert a character space separator between both string in order to avoid that $a_n$and $b_1$become a single token.

Lets assume that we have a function $card(.)$ that takes an unordered list of tokens and returns some kind of text cardinality. The Jaccard coefficient can be rewritten as.

$$Jaccard(A, B) = \frac{card(A) + card(B) - card(A \oplus B)}{card(A \oplus B)} \tag{3.6}$$

The function $card(.)$ can have a crisp behavior counting the number different tokens in the multi-token string. This function has to satisfies two basic properties:

$$card(a_1, a_2, ..., a_n) = 1, \text{ if } (a_1 = a_2 = ... = a_n) \tag{3.7}$$

$$card(a_1, a_2, ..., a_n) = n, \text{ if } (a_1 \neq a_2 \neq ... \neq a_n) \tag{3.8}$$

$$card(a_1) = 1 \tag{3.9}$$

Intuitively, a soft-version of the function $card(.)$ for sets of tokens could consider the similarity between tokens to estimate their cardinality. For instance, the cardinality of a set of tokens with high similarity among them (e.g.{"Jimenez", "Gimenez", "Jimenes"}) should be closer to 1 than to 3. In the other hand, if the tokens have low similarity among them (e.g. {"giveaway", "girlfriend", "gingerbread"}) should be closer to 3 (i.e. n). In fact, tokens that do not share characters among them (e.g. {"Bush", "Clinton", "Kerry"}) the cardinality should be exactly 3. Therefore, a soft-version of the function $card(.)$ would depend on the pairwise similarities of the tokens.

Table 3.3: Token pairwise similarities with *edit-distance*-based similarity measure for "Sergio Sergio Sergio"

	Sergio	Sergio	Sergio
Sergio	1	1	1
Sergio	1	1	1
Sergio	1	1	1

Table 3.4: Token pairwise similarities with *edit-distance*-based similarity measure for "Jimenez Gimenez Jimenez"

	Jimenez	Gimenez	Jimenes
Jimenez	1	0.8571	0.8571
Gimenez	0.8571	1	0.7143
Jimenes	0.8571	0.7143	1

Lets consider the following three cardinality functions where $sim(a_i, a_j)$ is a similarity measure between a_i and a_j, normalized in the range [0,1]:

$$card_A(a_1, a_2, ..., a_n) = \sum_{i=1}^{n} \left[\frac{1}{\sum_{j=1}^{n} sim(a_i, a_j)} \right] \tag{3.10}$$

$$card_B(a_1, a_2, ..., a_n) = 1 + \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} sim(a_i, a_j) \tag{3.11}$$

The time complexity of both $card_A(.)$ and $card_B(.)$ is $O(n^2)$. In order to use some resemblance coefficient to compare two multi-token strings A and B with m and n tokens respectively, it is necessary to compute $card(A)$, $card(B)$ and $card(A \oplus B)$. The total time complexity of compute the three cardinalities is $O((m+n)^2)$ multiplied by the time complexity of the function $sim(a, b)$. Note that the $card_A(.)$ function reproduces the behavior of the standard cardinality in the classic set theory when the internal function $sim(a_i, a_j)$ is a crisp string comparator.

Now, lets consider some examples. In tables 3.3, 3.4, 3.5 and 3.6 the token pairwise character-based similarities measured with the standard *edit-distance* are shown for the following sample strings: "Sergio Sergio Sergio", "Jimenez Gimenez Jimenes", "giveaway girlfriend gingerbread" and "Bush Clinton Kerry".

Now, the table 3.7 shows the calculations of the three proposed cardinality functions in equations 3.10 and 3.11.

The proposed concept of cardinality in set of tokens is similar to the size of the compressed object information information used in the Compression Distance (see sub-section 3.2.1.5). In fact, the function $card(.)$ can be viewed as an estimation of the size of the compressed information within the string. Now consider the following example comparing two multi-token strings with the

Table 3.5: Token pairwise similarities with *edit-distance*-based similarity measure for "giveaway girlfriend gingerbread"

	giveaway	girlfriend	gingerbread
giveaway	1	0.1999	0.3636
girlfriend	0.1999	1	0.4545
gingerbread	0.3636	0.4545	1

Table 3.6: Token pairwise similarities with *edit-distance*-based similarity measure for "Bush Clinton Kerry"

	Bush	Clinton	Kerry
Bush	1	0	0
Clinton	0	1	0
Kerry	0	0	1

Table 3.7: Examples of cardinality estimation with functions $card_A(.)$ and $card_B(.)$.

Multi-token string example	$card_A$	$card_B$
"Sergio Sergio Sergio"	1	1
"Jimenez Gimenez Jimenes"	1.1462	1.3809
"giveaway girlfriend gingerbread"	1.7939	2.3212
"Bush Clinton Kerry"	3	3

Dice coefficient, the cardinality function $card_A(.)$ and edit-distance as internal character-level similarity measure:

A ="Jimenez Vargas Sergio Gonzalo"

B ="Sergio Gonzalez Gimenez V."

$card_A(A)$= 3.132616487

$card_A(B)$ =3.454545455

$card_A(A \oplus B)$=3.867153978

$Dice(A,B) = \frac{2 \times (card_A(A) + card_A(B) - card_A(A \oplus B))}{card_A(A) + card_A(B)}$

$Dice(A,B) = 0.825851251$

The combination approaches previously presented (i.e. *Monge-Elkan* and *Token Edit-Distance*) that returns a measure of comparison between to multi-token strings A and B, uses the internal $sim(a,b)$ function only to compare tokens belonging to A and B. Tokens belonging to the same string never are being compared. In the same way, the character-level measures only make comparisons between the characters of both strings. The method proposed in this section firstly compares the tokens of each string between them when $card(A)$ and $card(B)$ are computed. Further, the combination of the pairwise comparison of the tokens of $A \oplus B$ is used to compute $card(A \oplus B)$. This additional information have not been used in the past, the question to be answered is: can this information increase the performance of a string measure in a string matching problem?

The most similar approach to ours have been proposed recently by Michelson and Knoblock [58]. They combined the Dice coefficient $2 \times |A \cap B|/(|A| + |B|)$ with the Jaro-Winkler measure (see section 3.2.1.2) in order to compare two multi-token strings A and B. Michelson and Knoblock established a threshold $\theta_{internal}$ for the Jaro-Winkler measure. Next, two tokens are put into the intersection $A \cap B$ if those two tokens have a Jaro-Winkler similarity above $\theta_{internal}$. This approach has the same drawback of the soft TD-IDF [25], which is the necessity of a second threshold for the internal measure additional to the threshold that is required for the combined method in order to decide the matching.

3.6 Experimental Evaluation

The aim of this section is to compare the performance of a set of character-level string similarity measures compared with new measures obtained from the combination of a character-level measure with some token-level measure. This comparison aims to answer the following question: is it better to tokenize (i.e. separate into words) the text strings and treat them as sequences of tokens compared with to treat the text strings as simple sequences of characters?

Table 3.8: Data-sets used to carry out experiments.

Name	#records	$\|rel1\|$	$\|rel2\|$	$\|rel1\| \times \|rel2\|$	#matches	#tokens
Birds-Scott1*[2]	38	15	23	345	15	122
Birds-Scott2*	719	155	564	87,420	155	3,215
Birds-Kunkel*	336	19	315	5,985	19	1,302
Birds-Nybird*	985	67	918	61,506	54	1,957
Business*	2,139	976	1,163	1,135,088	296	6,275
Game-Demos*	911	113	768	86,784	49	3,263
Parks*	654	258	396	102,168	250	2,119
Restaurants*	863	331	532	176,062	112	9,125
UCD-people*	90	45	45	2,025	45	275
Animals*	1,378	689	689	474,721	327	3,436
Hotels[3]	1,257	1,125	132	148,500	1,028	9,094
Census[4]	841	449	392	176,008	327	4,083

Additionally, the new combination methods presented in sections 3.4 and 3.5 will be compared with two popular combination methods in order to assess the value of those new approaches.

3.6.1 Experimental Setup

The comparison of the performance of various character- and token-level string comparison measures has been made in the past on the name matching task [25, 21, 73, 12]. The name matching task consist of compare two multi-token strings that contains names, addresses, telephones and other information, and decide if the pair of strings refers to the same entity of not.

3.6.1.1 Data-sets

The data-sets for string matching are usually divided in two relations. A specific subset of the Cartesian product of those two relations is the set of valid matches. The table 3.8 describes the 12 used data-sets with the total number of strings, how those strings are divided in the two relations ($rel1$ and $rel2$), the number of pair that are valid matches (set of valid matches) the size of the Cartesian product of the relations and the total number of tokens. The *Animal* data-set was not originally separated in two relations. The relations were obtained using the 327 valid matches and the 689 strings involved in those matches.

The tokenizing process was made using as token (word) separator the following characters: *space*, '*(*', '*)*', '=', '-', '/',*coma* , '*;*' and '*:*'. Additionally, all consecutive double token separators were withdrew and all heading and trailing blank spaces were removed. The number of tokens obtained with this tokenizing policy in each data-set is reported in the last column in Table 3.8. Finally, all characters are converted to uppercase characters and some characters with french accents were replaced with their equivalent characters without accent. In

Table 3.9: Data-set sample matches.

Data-set(rel#)	Sample String
Birds-Scott1(1)	"Great Blue Heron"
Birds-Scott1(2)	"Heron: Great blue (Grand Héron) Ardea herodias"
Birds-Scott2(1)	"King Rail Rallus elegans"
Birds-Scott2(2)	"Rail: King (Rale élégant) Rallus elegans"
Birds-Kunkel(1)	"Eastern Towhee"
Birds-Kunkel(2)	"Rufous-sided towhee (Pipilo erythrophthalmus)"
Birds-Nybird(1)	"Yellow-rumped Warbler"
Birds-Nybird(2)	"Yellow-rumped ("Myrtle") Warbler"
Business(1)	"BBN Corporation"
Business(2)	"BOLT BERANEK & NEWMAN INC"
Game-Demos(1)	"Learning in Toyland"
Game-Demos(2)	"Fisher-Price Learning in Toyland"
Parks(1)	"Timucuan Ecol. & Hist. Preserve"
Parks(2)	"Timucuan Ecological and Historic Preserve"
Restaurants(1)	"Shun Lee West 43 W. 65th St. New York 212/371-8844 Asian"
Restaurants(2)	"Shun Lee Palace 155 E. 55th St. New York City 212-371-8844 Chinese"
UCD-people(1)	"Barragry, Thomas B."
UCD-people(2)	"Dr. Thomas B. Barragry"
Animals(2)	"Amargosa pupfish"
Animals(2)	"Pupfish, Ash Meadows Amargosa"
Hotels(1)	"3* Four Points Sheraton Airport 3/13 $30"
Hotels(2)	"Four Points Barcelo by Sheraton Pittsburgh 3* Airport PIT"
Census(1)	"GASBARRO PABLOS U 401 VILLAGRANDE"
Census(2)	"GASBARRD PABLO V 401 VILLAGRANDE"

table 3.9for the sake of illustration a sample strings (in some cases hard-to-match strings) is shown for each data-set.

3.6.1.2 Experimental Setup

Experiments were carried out comparing the full Cartesian product between the relations $rel1$ and $rel2$ without using any blocking strategy. This is a time-consuming approach but give us the complete elements to compare the performance of different matching techniques. The baseline experiments consist of use a character-based string measure (e.g. *edit-distance, 2-grams, Jaro-distance*) and compare the full length strings with the measure. In order to perform a fairer comparison, the baseline experiments were not carried out with the original strings but with the tokenized string re-joined with the character space as only separator. The string metrics used in the baseline experiments are shown with its names as follows:

1. *exactMatch*: It compares two strings returning 1 if are identical and 0 if

not.

2. *ed.Dist*: It returns the a normalized similarity using the expression: $1 - \frac{edit-distance(A,B)}{max(|A|,|B|)}$. The edit-distance used the three basic edit operations (i.e. *insertion*, *deletion* and *replacement*) at cost 1.

3. *Jaro*: It returns the Jaro-distance metric explained in sub-section 3.2.1.2 that is in fact a similarity measure.

4. *2grams*: It returns the ratio $\frac{\#commonBigrams(A,B)}{max(|A|,|B|)}$. The common bi-grams were counted considering padding characters as it was explained in sub-section 3.2.1.3.

The previously listed measures will also be used as internal similarity measure $sim(a, b)$ in several token combination methods. The first set of combination methods are listed next.

1. *SimpleStr.*: Not combination at all, only compares the strings with the internal $sim(a, b)$ measure.

2. *MongeElkan*: The combination method proposed by Monge and Elkan using equation 3.1 (means $m = 1$).

3. *MongeElkan0*: Extension to the basic Monge-Elkan method with exponent $m = 0.00001$ in equation 3.4.

4. *MongeElkan0.5*: Extension to the basic Monge-Elkan method with exponent $m = 0.5$ in equation 3.4.

5. *MongeElkan1.5*: Extension to the basic Monge-Elkan method with exponent $m = 1.5$ in equation 3.4.

6. *MongeElkan2*: Extension to the basic Monge-Elkan method with exponent $m = 2$ in equation 3.4.

7. *MongeElkan5*: Extension to the basic Monge-Elkan method with exponent $m = 5$ in equation 3.4.

8. *MongeElkan10*: Extension to the basic Monge-Elkan method with exponent $m = 10$ in equation 3.4.

9. *max_max*: Equivalent to the extension of Monge-Elkan method with $m \to \infty$, i.e. $sim_{max}(a, b) = \max_{i=1}^{|a|} \max_{j=1}^{|b|} sim(a_i, b_j)$.

10. *TokenEd.Dist*: The edit-distance metric at token-level where the token replacement cost is $sim(a, b)$ and insertion an deletion cost are set at 1.

Additionally to the first set of combination methods, a second set with resemblance coefficients using the soft text cardinalities $card_A(.)$ and $card_B(.)$ (equations 3.10 and 3.11 respectively) are listed and explained in table 3.10.

Table 3.10: Second set of character-/Token-level combination methods based in cardinality

#	Name	Description
11	*Jaccard*	$\frac{card(A \cap B)}{card(A \cup B)}$
12	*Dice*	$\frac{2 \times card(A \cap B)}{card(A)+card(B)}$
13	*cosine*	$\frac{card(A \cap B)}{\sqrt{card(A) \times card(B)}}$
14	*Sorensen*	$\frac{2 \times card(A \cap B)}{card(A \cup B)+card(A \cap B)}$
15	*compression*	$\frac{card(A \cup B)-\min(card(A),card(B))}{\max(card(A),card(B))}$
16	*overlap*	$\frac{card(A \cap B)}{\min(card(A),card(B))}$
17	*R1*	$\frac{card(A \cap B)}{\max(card(A),card(B))}$
18	*R8*	$\frac{\max(card(A),card(B))}{card(A \cup B)}$
19	*R12*	$\frac{\min(card(A),card(B))}{card(A \cup B)}$

The total number of combination strategies at token level are the addition of the 10 methods previously listed plus the 9 cardinality-based methods in table 3.10. The 9 cardinality-based methods were used with the two cardinality functions $card_A(.)$ and $card_B(.)$ (see equations 3.10and 3.11), that is 28 token combination methods in total. Those 28 token-based methods were combined with the four character-based measures in table **??** for a total of 112 multi-token comparison measures. Each combination will be referred using the names before their definitions ant the column *Name* in table 3.10. The format of the combined method names is *name_of_character_level_metric* (hyphen) *name_of_token_level_metric* (*cardinality_function*). For instance, a method that uses the *edit-distance* as internal inter-token measure and the *overlap* coefficient with the cardinality function $card_B(.)$ is referred as "ed.dist.-overlap(B)". The combination methods in table **??** are referred without the suffix "(A)" of "(B)".

We define an experiment as a pair {*string_matching_method*, *data-set*}. The total number of experiments carried out is the test of all the 112 string matching combined methods in the 12 proposed data-sets, that is 1344 experiments. For each experiment, the string similarity measure is computed and stored for each pair strings in the Cartesian product of the two relations of the data-set. All computed similarities were in the range [0,1] having 1 as the maximum level of similarity and 0 as the maximum dis-similarity.

3.6.1.3 Experiments Performance Measure

The matching problem between two relations (sets) can be viewed as a classification problem over the Cartesian product of the sets where a subset of this Cartesian product is the set of valid matches (i.e. *positives*) and its complement are non-valid matches (i.e. *negatives*). For each experiment, a similarity

measure value in the range [0,1] is computed for each possible string pair in a data-set. However, in order to decide if a pair is a valid match it is necessary to establish a threshold θ. Similarity values greater than or equal to θ are labeled as *positive* and as *negative* in the alternative case. For a specific value of θ it is possible to define:

True Positive: a string pair in the set of valid matches labeled as *positive.*

False Positive: a string pair NOT in the set of valid matches labeled as *negative.*

False Negative: a string pair in the set of valid matches labeled as *negative.*

The classification performance in two-class problems are usually measured using *precision*, *recall* and *F-measure* that is the harmonic mean between precision and recall.

$$precision = \frac{\#true\,positives}{\#true\,positives + \#false\,positives}$$

$$recall = \frac{\#true\,positives}{\#true\,positives + \#false\,negatives}$$

$$F - measure = \frac{2 \times precision \times recall}{(precision + recall)}$$

For each experiment (i.e. a pair $\{string\_matching\_method, data\text{-}set\}$) the counting of true positives, false positives and false negatives were made ranging θ form 0 to 1 in steps of 0.01. Consequently, for each experiment varying θ, 101 values of precision, recall and F-measure are obtained. The graph shown in figure 3.2 plots the data obtained for the experiment {*ed.Dist.-Cosine(A)* ,*Business*}. The graph also shows the trade-off between precision and recall varying the similarity threshold θ. Recall has decreasing tendency whereas precision is increasing. The maximum value of the F-measure known as *F1-score* is obtained in the threshold θ with the best balance between precision and recall. F1-score can also be seen as a general performance measure of the experiment. The F1-score are closer to the point where the curves of precision and recall intersect, that is in fact another general performance measure. However, the F1-score measure is used more commonly.

The same data can be used to plot a precision-recall curve. The figure 3.3 shows precision-recall curve obtained from the experiment {*ed.Dist.-Cosine(A)* ,*Business*}. Using this curve it is possible to obtain another general performance measure called *interpolated average precision* (IAP) which is commonly used in information retrieval problems [6]. One method to compute the IAP measure is interpolate the precision-recall curve at 11 evenly separated recall points (i.e. 0.0, 0.1, 0.2, ...,1.0}. Interpolated precision at recall point r is the maximum

Figure 3.2: Precision, recall and F-measure vs. threshold curves for the experiment {*ed.Dist.-Cosine(A)* ,*Business*}.

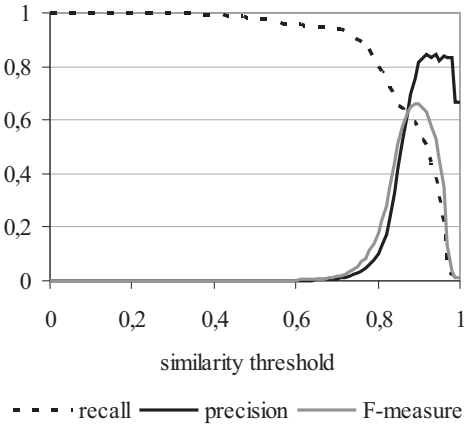


Figure 3.3: Precision-recall curve for the experiment {*ed.Dist.-Cosine(A)* ,*Business*}.

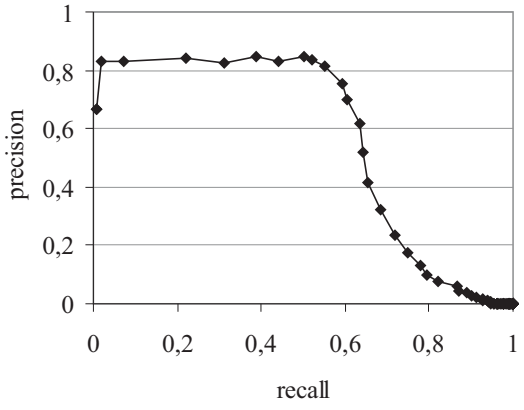
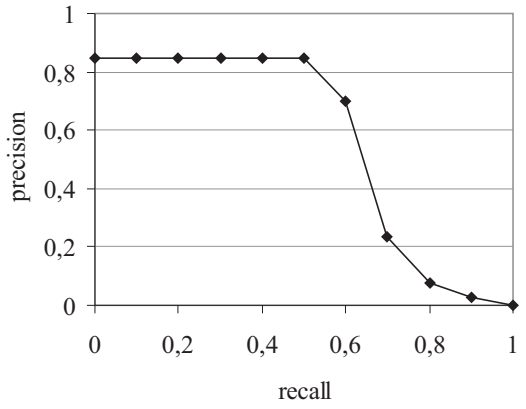


Figure 3.4: Interpolated precision-recall curve at 11 evenly separated recall points



precision obtained at any point with recall greater than or equal to r. Figure 3.4 shows an interpolated version of the curve in figure 3.3 using the described procedure.

The are under the the interpolated precision-recall curve is a general performance measure of the experiment. Consider a classifier that obtains a precision and recall values of 1 at some θ. The interpolated precision-recall curve becomes a step (i.e. 1×1 square) with its vertex at the point (1,1). The area under that curve is 1 corresponding to the maximum possible performance. In order to compute that area under the curve the 11 evenly separated precision values are averaged. The measure obtained is known as *interpolated average precision* (IAP). The IAP reflects the general performance of the matching measure but not the performance of the matching task. The later depends of the selected threshold θ, which is unknown. We are interested in to measure the performance of the matching method. Clearly, the better the performance of the matching method, the better the performance of the matching task at reasonable θ values.

The IAP were selected as the primary performance measure since its graphical interpretation (area under the curve) that make it more convenient for operations of aggregation such as the average. For instance, in order to report in only one number the performance of one of the 112 matching methods, its values of IAP obtained for the 12 data-sets are averaged and reported with the name *avg. IAP*.

3.6.1.4 Statistical Test

For each one of the tested 112 matching methods the average IAP over the 12 data-set is a performance measure. Given any pair of matching methods m_1

and m_2, if the average IAP obtained in m_1 is greater than the one of m_2, is it possible to say that m_1 outperforms m_2? In order to answer this question the *Wilcoxon Matched Pairs Signed-Ranks Test* [91]was used.

The Wilcoxon rank-sum test is a nonparametric alternative to the two sample *t-test*. The term non-parametric generally means that there is no assumption that the underlying population has a Normal distribution. This test evaluates whether or not the median of two paired samples are the same (H_0 null hypothesis). We are interested in a one tailed test, where the alternative hypothesis is that the median of the results of m_1 is greater than the results of m_2.

The assumptions required by the test is that the results obtained with the methods m_1 and m_2 have identical distributions. If the pairs are assumed to have identical distributions, their differences should always have a symmetrical distribution that is in fact the more important assumption of the test. Also, it is required that the sample has been randomly chosen from the population (he sample in our case is the 12 used data-sets). Additionally, this test requires no assumption that you had large samples.

The procedure to perform the test is the following one:

1. Calculate the difference between each pair of results from the sample.
2. Ignore the zero differences (we ignored differences lower than 0.0001). Clearly ignored differences reduces the size of the sample n.
3. Rank the absolute values of the remaining differences. If there are identical differences average their ranks.
4. Sum the ranks of the positive differences, W_+ , and sum the ranks of the negative differences W_-.
5. Compare the test statistics W_- and compare with the critical values in the tables. If W_- is lower than or equal to the critical value for n in table 3.11, then reject the null hypothesis.

The critical values used for the Wilcoxon test statistic W for different sample sizes n are show in table 3.11.

Now, lets consider the results of the experiments for the methods *2gram-MongeElkan2* and *2gram-compress(C)* shown in the first three columns in table 3.12. The value of the test statistic W_- is 4, the critical value for $n = 12$ in table 3.11 with a significance of 2.5% is 14. Thus, the null hypothesis is rejected and the evidence shows, at the 2.5% significance level, that the matching method *2gram-MongeElkan2* outperforms *2gram-compress(C)*.

Another example comparing *2gram-MongeElkan2* and *2gram-compress(C)* is shown in table 3.13. The value of the test statistic W_- is 28 and the critical value is 14 at the significance level of 2.5%. Thus, the null hypothesis is accepted and although the value of the average IAP is greater for *2gram-MongeElkan2* there is not enough statistical evidence to affirm that this method outperforms *2gram-compress(A)*.

Table 3.11: Critical values for the Wilcoxon test statistic for one-tailed test.

	Significance		
n	10.0%	5.0%	2.5%
5	2	1	-
6	3	3	1
7	5	4	3
8	8	6	4
9	11	9	6
10	13	11	9
11	17	14	11
12	21	18	14
13	26	22	17
14	31	26	18
15	36	31	22

Table 3.12: IAP Wilcoxon sign test for experiments *2gram-MongeElkan2* (method1) and *2gram-compress(C)*(method2)

Data-set	method1	method2	diff.	rank	W_+	W_-
Birds-Nybirds	0.694809	0.674526	0.020283	1	1	-
Game-Demos	0.541242	0.491103	0.050139	2	2	-
Animals	0.128630	0.074987	0.053643	3	3	-
Census	0.652203	0.724021	-0.071818	4	-	4
Birds-Scott1	0.869519	0.781612	0.087907	5	5	-
Restaurants	0.891184	0.781859	0.109325	6	6	-
UCD-people	0.849196	0.697854	0.151343	7	7	-
Parks	0.839082	0.654155	0.184927	8	8	-
Birds-Scott2	0.877792	0.690973	0.186820	9	9	-
Hotels	0.551949	0.363171	0.188778	10	10	-
Birds-Kunkel	0.892562	0.697015	0.195547	11	11	-
Business	0.717120	0.498934	0.218186	12	12	-
AVERAGE	0.708774	0.594184		$\sum$	74	4

Table 3.13: IAP Wilcoxon sign test for experiments *2gram-MongeElkan2* (method1) and *2gram-compress(A)* (method2)

Data-set	metthod1	method2	diff.	rank	W_+	W_-
Restaurants	0.891184	0.890231	0.000953	1	1	-
Birds-Scott1	0.869519	0.858471	0.011048	2	2	-
Parks	0.839082	0.851435	-0.012353	3	-	3
Birds-Nybird	0.694809	0.729448	-0.034639	4	-	4
Hotels	0.551949	0.508777	0.043172	5	5	-
Animals	0.128630	0.083351	0.045280	6	6	-
Business	0.717120	0.671056	0.046064	7	7	-
Birds-Scott2	0.877792	0.779647	0.098145	8	8	-
UCD-people	0.849196	0.738955	0.110241	9	9	-
Census	0.652203	0.806547	-0.154344	10	-	10
Game-Demos	0.541242	0.708952	-0.167709	11	-	11
Birds-Kunkel	0.892562	0.460259	0.432303	12	-	12
AVERAGE	0.708774	0.673927		$\sum$	50	28

The Wilcoxon's test were performed to compare each one of the 112 matching methods against the other 111 methods. For each method, the quotient of the number of methods that outperformed it, divided by 111 was reported as the *Wilcoxon's Rate*.

3.6.2 Results

3.6.2.1 General Ranking

The tables 3.14, 3.15 and 3.16 show an ordered list by *avg. IAP* with the results obtained for each one of the 112 combination matching methods tested. The *avg. IAP* values were obtained averaging the IAP results obtained in the 12 data-sets for each matching method. The column *test* shows the number of times that the method statistically outperformed compared with the other 111 methods using the Wilcoxon's test. That is the number of accepted alternative hypothesis that the median of the results obtained with the method is greater than the median of the method with which it is being compared with significance of 5%. Finally, the last column shows the rate of passed statistical test divided by the number of methods that the method outperformed. Namely:

$$Wilcoxon'sRate = \frac{test}{112 - rank}$$

It is important to note the obtained ranks by the baseline methods. That is *2gram*, *Jaro*, *ed.Dist.* and *exactMatch* respectively: 19, 65, 70 and 97. The case of the *exactMatch* (rank=97) comparison method is particular, which obtained and IAP of 0.1835057 surpassing other 15 methods. However, as it was expected, this method did not outperformed statistically other methods. The complete

rank list is reported for the sake of completeness and to give a general guide of the "good" and the "not as good" method combinations.

3.6.2.2 Comparison versus Baselines and the MongeElkan Method

The baselines of ours experiments are four character-level measures comparing pairs of multi-token strings as simple sequences of characters. The four measures are: 2-grams, exact-match, edit-distance and Jaro-distance. The table 3.17 compares the results of the baseline methods with the procedure of separate the string in tokens (words), compare the tokens with the Monge-Elkan method (see equation 3.1) using as internal measure $sim(a, b)$ the same baseline method. For instance, the baseline result reported in the first row correspond to the method *2gram-simpleStr.* and the column named MongeElkan correspond to the results for the method *2gram-MongeElkan*.

The four baseline measures combined with the Monge-Elkan method improved the IAP compared with the baseline in all cases; the relative improvement is reported in the last column. The column W_- shows the test statistic for the one-tailed Wilcoxon's test. The critical value of W_- for $n = 12$ with significance of 0.05 is 18 and approximately 21 with significance of 0.1. So, the only improvement with enough statistical evidence is the obtained with *exactMatch-MongeElkan* versus *exactMatch-simpleStr* (starred with * in the table 3.17). The Monge-Elkan combination method is considered by the string processing community as one of the best static combination methods. However, the obtained improvements there were not enough statistical evidence of consistency in the sample of 12 data-sets. This result also shows that the Wilcoxon's test is not as easy to pass even with a relative high significance.

The results in table 3.18 have the same format as in table 3.17 but additionally report in the column "Best Method" the name of the combination method with the highest avg. IAP reached for each baseline measure. The starred with one * and two ** mean that the Wilcoxon's test was passed with significance of 0.05 and 0.1 respectively. Similarly, the baseline methods were compared with the best methods for each baseline measure, the results are shown in table 3.19.

3.6.2.3 Results for Extended Monge-Elkan Method

The extension proposed for the Monge-Elkan method in the equation 3.4 were tested for values of the exponent m in 0.00001, 0.5, 1.0 (standard MongeElkan), 1.5, 2.0, 5.0 and 10.0. The averaged IAP considering the 12 data-sets for each internal character-level similarity measure is plotted in figure 3.5. The average of the four curves is shown in figure 3.6. Additionally, the Wilcoxon's test was used to compare the standard MongeElkan method ($m = 1$) versus the extended method with exponents m with the values of 1.5, 2.0 and 5.0. The aim of this test is to establish if there are statistical evidence that if the standard MongeElkan method is improved when the exponent m in the equation 3.4 is above 1. The table 3.20 shows the results of the Wilcoxon's test statistic W_-. The value of n was 12 for all three comparisons, so the critical values for W_- were 21 and

Table 3.14: General ranking for combined string matching methods. Part 1/3

rank	method	avg. IAP	tests	Wilcoxon's Rate
1	*2gram-cosine(A)*	0.739927	103	92.79%
2	*2gram-Jaccard(A)*	0.729502	94	85.45%
3	*2gram-Dice(A)*	0.728885	94	86.24%
4	*2gram-Sorensen(A)*	0.728885	94	87.04%
5	*exactMatch-cosine(A)*	0.714650	80	74.77%
6	*2gram-MongeElkan1.5*	0.710334	73	68.87%
7	*2gram-MongeElkan2*	0.708774	83	79.05%
8	*exactMatch-Jaccard(A)*	0.707249	75	72.12%
9	*exactMatch-Sorensen(A)*	0.707147	75	72.82%
10	*exactMatch-Dice(A)*	0.707147	75	73.53%
11	*Jaro-MongeElkan5*	0.704047	70	69.31%
12	*ed.Dist.-MongeElkan2*	0.703136	67	67.00%
13	*2gram-MongeElkan*	0.701698	70	70.71%
14	*ed.Dist.-MongeElkan1.5*	0.696779	69	70.41%
15	*Jaro-MongeElkan10*	0.694706	74	76.29%
16	*ed.Dist.-MongeElkan5*	0.689713	69	71.88%
17	*2gram-overlap(A)*	0.681110	57	60.00%
18	*ed.Dist.-MongeElkan*	0.678082	61	64.89%
19*	*2gram-simpleStr.*	0.677859	51	54.84%
20	*2gram-MongeElkan5*	0.677835	62	67.39%
21	*exactMatch-compress(A)*	0.674830	53	58.24%
22	*exactMatch-R1(A)*	0.674830	53	58.89%
23	*2gram-compress(A)*	0.673927	54	60.67%
24	*2gram-R1(A)*	0.673825	54	61.36%
25	*exactMatch-overlap(A)*	0.672695	54	62.07%
26	*ed.Dist.-cosine(A)*	0.672688	56	65.12%
27	*Jaro-MongeElkan2*	0.671519	57	67.06%
28	*ed.Dist.-MongeElkan10*	0.662875	56	66.67%
29	*2gram-MongeElkan10*	0.658941	54	65.06%
30	*exactMatch-MongeElkan*	0.658511	53	64.63%
31	*exactMatch-MongeElkan2*	0.658385	53	65.43%
32	*exactMatch-MongeElkan0.5*	0.657882	53	66.25%
33	*exactMatch-MongeElkan1.5*	0.657856	53	67.09%
34	*exactMatch-MongeElkan5*	0.657585	53	67.95%
35	*exactMatch-MongeElkan10*	0.656911	53	68.83%
36	*ed.Dist.-Jaccard(A)*	0.656911	49	64.47%
37	*2gram-MongeElkan0.5*	0.656879	50	66.67%
38	*Jaro-MongeElkan1.5*	0.656635	50	67.57%
39	*ed.Dist.-Sorensen(A)*	0.656062	49	67.12%
40	*ed.Dist.-Dice(A)*	0.656062	49	68.06%

Table 3.15: General ranking for combined string matching methods. Part 2/3

rank	method	avg. IAP	tests	Wilcoxon's Rate
41	*Jaro-MongeElkan*	0.636822	46	64.79%
42	*ed.Dist.-MongeElkan0.5*	0.635641	46	65.71%
43	*2gram-cosine(B)*	0.629773	42	60.87%
44	*2gram-tokenEd.Dist.*	0.629443	35	51.47%
45	*2gram-R8(A)*	0.628162	39	58.21%
46	*2gram-Dice(B)*	0.624980	42	63.64%
47	*2gram-Sorensen(B)*	0.624980	42	64.62%
48	*exactMatch-cosine(B)*	0.616827	42	65.63%
49	*Jaro-MongeElkan0.5*	0.616575	40	63.49%
50	*2gram-Jaccard(B)*	0.615958	39	62.90%
51	*exactMatch-Sorensen(B)*	0.613965	41	67.21%
52	*exactMatch-Dice(B)*	0.613965	41	68.33%
53	*Jaro-MongeElkan0*	0.610215	36	61.02%
54	*exactMatch-overlap(B)*	0.609747	39	67.24%
55	*exactMatch-Jaccard(B)*	0.600628	41	71.93%
56	*2gram-overlap(B)*	0.596793	39	69.64%
57	*ed.Dist.-tokenEd.Dist.*	0.594873	31	56.36%
58	*ed.Dist.-MongeElkan0*	0.594511	33	61.11%
59	*2gram-R1(B)*	0.594198	39	73.58%
60	*2gram-compress(B)*	0.594184	39	75.00%
61	*exactMatch-compress(B)*	0.592150	39	76.47%
62	*exactMatch-R1(B)*	0.591914	39	78.00%
63	*exactMatch-R8(A)*	0.591896	33	67.35%
64	*exactMatch-tokenEd.Dist.*	0.585316	31	64.58%
65*	*Jaro-simpleStr.*	0.577744	31	65.96%
66	*ed.Dist.-compress(A)*	0.573967	32	69.57%
67	*ed.Dist.-R1(A)*	0.573967	32	71.11%
68	*2gram-MongeElkan0*	0.554194	30	68.18%
69	*2gram-R14(A)*	0.553339	32	74.42%
70*	*ed.Dist.-simpleStr.*	0.550062	31	73.81%
71	*ed.Dist.-cosine(B)*	0.536572	31	75.61%
72	*ed.Dist.-Sorensen(B)*	0.536445	31	77.50%
73	*ed.Dist.-Dice(B)*	0.536445	31	79.49%
74	*exactMatch-R14(A)*	0.523569	30	78.95%
75	*ed.Dist.-Jaccard(B)*	0.518179	30	81.08%
76	*ed.Dist.-R14(A)*	0.516899	30	83.33%
77	*ed.Dist.-R1(B)*	0.512050	30	85.71%
78	*ed.Dist.-compress(B)*	0.511994	30	88.24%
79	*ed.Dist.-overlap(B)*	0.500791	26	78.79%
80	*ed.Dist.-overlap(A)*	0.466801	28	87.50%

Table 3.16: General ranking for combined string matching methods. Part 3/3

rank	method	avg. IAP	tests	Wilcoxon's Rate
81	*ed.Dist.-R8(A)*	0.450146	27	87.10%
82	*Jaro-tokenEd.Dist.*	0.427008	28	93.33%
83	*2gram-R14(B)*	0.341112	24	82.76%
84	*exactMatch-MongeElkan0*	0.334419	16	57.14%
85	*ed.Dist.-R14(B)*	0.317554	21	77.78%
86	*exactMatch-R14(B)*	0.304862	19	73.08%
87	*Jaro-compress(A)*	0.255566	15	60.00%
88	*Jaro-R1(A)*	0.255534	15	62.50%
89	*Jaro-R14(A)*	0.252449	14	60.87%
90	*Jaro-Sorensen(B)*	0.218827	16	72.73%
91	*Jaro-Dice(B)*	0.218827	16	76.19%
92	*Jaro-cosine(B)*	0.217471	15	75.00%
93	*Jaro-Jaccard(B)*	0.216095	15	78.95%
94	*Jaro-R1(B)*	0.206173	14	77.78%
95	*Jaro-compress(B)*	0.206171	14	82.35%
96	*Jaro-R14(B)*	0.185595	7	43.75%
97*	*exactMatch-simpleStr.*	0.183505	0	0.00%
98	*Jaro-overlap(B)*	0.146254	11	78.57%
99	*Jaro-Jaccard(A)*	0.130602	8	61.54%
100	*Jaro-Sorensen(A)*	0.129052	6	50.00%
101	*Jaro-Dice(A)*	0.129052	6	54.55%
102	*Jaro-cosine(A)*	0.109623	3	30.00%
103	*Jaro-max_max*	0.101732	2	22.22%
104	*2gram-max_max*	0.101724	2	25.00%
105	*ed.Dist.-max_max*	0.101721	2	28.57%
106	*exactMatch-max_max*	0.101709	2	33.33%
107	*exactMatch-R8(B)*	0.064851	1	20.00%
108	*2gram-R8(B)*	0.061020	1	25.00%
109	*ed.Dist.-R8(B)*	0.060413	1	33.33%
110	*Jaro-R8(B)*	0.041945	0	0.00%
111	*Jaro-overlap(A)*	0.028704	1	100.00%
112	*Jaro-R8(A)*	0.028212	0	-

Table 3.17: Average IAP for baseline measures improved with MongeElkan method.

$sim(a,b)$	Baseline	*MongeElkan*	$W\_$	Improvement
2grams	0.677859(0.229198)	0.701698(0.227473)	33	3.52%
exactMatch	0.183505(0.274896)	0.658511(0.249272)	1*	258.85%
ed.Dist.	0.550062(0.296590)	0.678082(0.241435)	26	23.27%
Jaro	0.577744(0.289440)	0.636822(0.275805)	32	10.23%

Table 3.18: Average IAP for the Monge-Elkan combination method compared with the best combination method.

$sim(a,b)$	*MongeElkan*	Best Method	Best Method's avg.IAP	$W\_$	Improv.
2grams	0.701698(0.227473)	cosine(A)	0.739927(0.221174)	17*	5.45%
exactMatch	0.658511(0.249272)	cosine(A)	0.714650(0.245442)	18*	8.53%
ed.Dist.	0.678082(0.241435)	MongeElkan2	0.703136(0.224808)	19**	3.69%
Jaro	0.636822(0.275805)	MongeElkan5	0.704047(0.224379)	18*	10.56%

Table 3.19: Average IAP for baseline measures improved with the best combination method.

$sim(a,b)$	Baseline	Best Method	Best Method's avg.IAP	$W\_$	Improv.
2grams	0.677859(0.229198)	cosine(A)	0.739927(0.221174)	3*	9.16%
exactMatch	0.183505(0.274896)	cosine(A)	0.714650(0.245442)	0*	289.44%
ed.Dist.	0.550062(0.296590)	MongeElkan2	0.703136(0.224808)	21**	27.83%
Jaro	0.577744(0.289440)	MongeElkan5	0.704047(0.224379)	24	21.86%

18 for significance of 10% and 5%. The values that passed the Wilcoxon's text were starred with one or two * given its significance of 5% or 10% respectively.

3.6.2.4 Results of the New Cardinality-Based Methods

The presented results of this sub-section have the aim to assess the performance of the different tested resemblance coefficients. Once again, the average of the IAP results over the 12 data-sets and the four internal character-level string similarity measures (48 experiments in total) is computed for each resemblance coefficient. The tables 3.21a and 3.21b report the avg. IAP obtained using solely $card_A(.)$ or $card_B(.)$ respectively. The table 3.21 shows the same measure but using both cardinality function averaging 96 experiments in total.

3.6.2.5 Results by Data-set

In order to allow comparison with past and future studies, the method with best results obtained by data-set are reported in table 3.22 in the second and third columns. The majority of the methods with the best results involve new methods

Table 3.20: Values of the Wilcoxon's test statistic $W\_$ comparing extended Monge-Elkan method versus the standard Monge-Elkan method.

$sim(a,b)$	$W\_$ *MongeElkan1.5*	$W\_$ *MongeElkan2*	$W\_$ *MongeElkan5*
2grams	15*	19**	42
exactMatch	-	-	-
ed.Dist.	18*	20**	27
Jaro	5*	12*	12*

Figure 3.5: Average IAP behavior of the exponent m in extended Monge-Elkan method for each internal character-level string similarity measures.

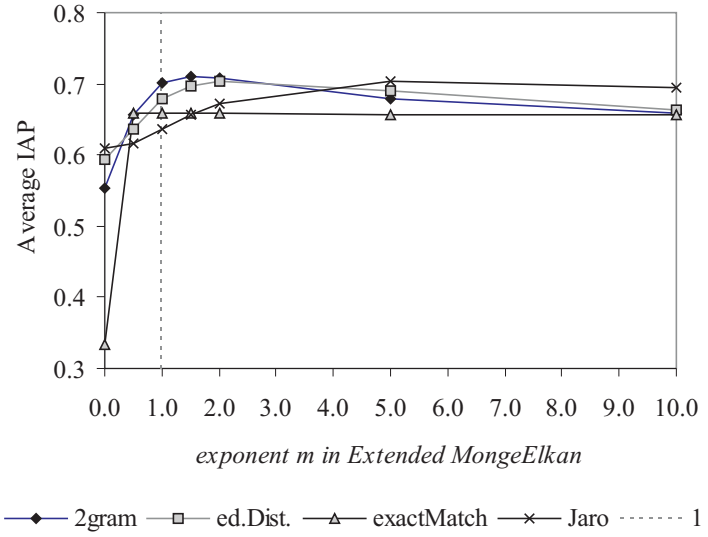


Figure 3.6: Average IAP behavior of the exponent m in extended Monge-Elkan method for all internal character-level string similarity measures (averaged).

Table 3.21: Ranks of resemblance coefficients performance in string matching.

a) $card_A(.)$

rank	token	avg.IAP(A)	std.dev.
1	cosine	0.559222	0.338731
2	Jaccard	0.556066	0.329359
3	Sorensen	0.555287	0.329435
4	Dice	0.555287	0.329435
5	compress	0.544573	0.286556
6	R1	0.544539	0.286581
7	overlap	0.462328	0.353293
8	R14	0.461564	0.267292
9	R8	0.424604	0.353888

b) $card_B(.)$

rank	token	avg.IAP(B)	std.dev.
1	cosine	0.500161	0.278444
2	Sorensen	0.498554	0.273099
3	Dice	0.498554	0.273099
4	Jaccard	0.487715	0.266854
5	compress	0.476125	0.256401
6	R1	0.476084	0.256355
7	overlap	0.463396	0.316394
8	R14	0.287280	0.212362
9	R8	0.057057	0.085837

c) $card_A(.)$ and $card_B(.)$

rank	token	avg.IAP(A&B)	std.dev.
1	cosine	0.529691	0.309846
2	Sorensen	0.526920	0.302332
3	Dice	0.526920	0.302332
4	Jaccard	0.521890	0.300131
5	compress	0.510349	0.272641
6	R1	0.510311	0.272634
7	overlap	0.462862	0.333582
8	R14	0.374422	0.255601
9	R8	0.240831	0.315805

Table 3.22: Best matching methods by data-set.

Data-set	1^{st} best method	avg. IAP	best baseline	avg. IAP	Improv.
Animals	*Jaro-MongeElkan1.5*	0.129343	*2gram*	0.066294	95.11%
Birds-Kunkel	*2gram-MongeElkan2*	0.892562	*Jaro*	0.773828	15.34%
Birds-Nybirds	*ed.Dist.-cosine(A)*	0.734960	*2gram*	0.714876	2.81%
Birds-Scott1	*exactMatch-cosine(A)*	0.889610	*2gram*	0.848485	4.85%
Birds-Scott2	*exactMatch-R8(A)*	0.909091	*ed.Dist.*	0.869423	4.56%
Business	*2gram-tokenEd.Dist.*	0.731611	*Jaro*	0.678614	7.81%
Census	*ed.Dist.*	0.859146	*ed.Dist.*	0.859146	-
Demos	*2gram-cosine(A)*	0.775548	*Jaro*	0.725928	6.84%
Hotels	*exactMatch-cosine(A)*	0.722460	*2gram*	0.610279	18.38%
Parks	*Jaro*	0.888874	*Jaro*	0.888874	-
Restaurants	*exactMatch-Sorensen(A)*	0.905571	*Jaro*	0.890970	1.64%
Ucd-people	*exactMatch-cosine(A)*	0.909091	*2gram*	0.796911	14.08%

or extended methods proposed in this thesis or combinations not previously testes in the past. Those best results were compared with the baseline method that obtained the best avg. IAP. Finally the relative improvement compared with the baseline are shown in the last column.

3.7 Conclusions

The final rank obtained from all the experiments (shown in tables 3.14, 3.15 and 3.16) shows and simultaneously hides information. Lets make an analogy between this ranking and the results of an athletic race. The four character-based metrics (*2gram*, *ed.Dist.*, *Jaro* and *exactMatch*) could be the nationalities of the competitors and the 28 combination methods could be performance-enhancing drugs (*simpleStr.* would correspond to fair competitors). The four fair competitors arrived at positions 19, 65, 70 and 97 between 112 participants. This bare result would show that doping methods clearly enhanced the performance of the athletes. However, the order in the rank list not necessarily reflected the performance of the athletes or the drugs. Consider the ranks 6 and 7, *2gram-MongeElkan1.5* reached a higher *avg. IAP* than *2gram-MongeElkan2* but had considerably less statistical evidence of its performance. Nevertheless, the "winner", *2gram-cosine(A)*, had also the highest statistical evidence.

The rank analysis is based in a comparison all-against-all and do not reflects a fair comparison against baselines. The tables 3.17, 3.18 and 3.19 showed specific comparisons against baseline measures. Additionally, the *MongeElkan* method was used as combination method baseline. We can conclude from those tables that both baselines (at character-level and at token-level) were outperformed. Even better the improvements obtained with the methods *cosine(A)*, *MongeElkan2* and *MongeElkan5* reached improvements statistically more consistent than the basic *MongeElkan* method. Those results showed clearly that

to treat text strings as sequences of tokens instead of character sequences has an important gain.

The conclusions that can be extracted from the empirical results are summarized as follows:

- The cardinality function $card_A(.)$ seems to be a good estimator of the "soft-cardinality" or "the size of the compressed information" of a set of words. Thus, this function in combination with resemblance coefficients provides a convenient general method for text string comparisons.
- The resemblance coefficients that have in their numerator the cardinality of the intersection of the tokens such as *cosine*, *Jaccard*, *Sorensen* and *Dice* seems to be suitable to the comparison of sets of words. Particularly, the *cosine* coefficient reached the best results and it had a good and consistent behavior. To the best of our knowledge, *cosine* have not used as resemblance coefficient in the past in the string matching task.
- The use of the generalized mean instead of the arithmetic mean in the MongeElkan measure seems to improve consistently the performance of the basic method when the exponent m is greater than 1 (equation 3.4).

Chapter 4

Knowledge-based Word Sense Disambiguation Methods for Term Disambiguation

In chapter 2 the laptop ontology, lexicon and corpus were presented, and in chapter 3 existing and new proposed techniques to approximately match the lexicon entries against the document corpus were compared. As a result, lexicon terms were identified in the token sequence of the documents in the corpus. However, a new problem rises: ambiguity.

The first source of ambiguity is *homoymy* in the lexicon. As it was showed in table 2.7, a term in the lexicon can be linked to different concepts. For instance, the acronym term "MB" can be linked to the terms "MegaByte" and "MotherBoard". In section 2.3 we concluded that the 17.06% of the terms in the lexicon have this kind of ambiguity.

The second ambiguity source is the *morphological similarity*. This ambiguity is originated by the fact that if an approximate string comparison technique is used to compare two "similar" strings, then they can be considered as equals. For instance, if the terms that are being compared with the edit distance using a threshold $\theta = 0.75$, strings such as "Blue-ray" and "Blue gray" are considered a valid match even though the former is an optical disk technology and the former is a laptop exterior color. However, it is expected that the use of approximate string comparison techniques provides resilience against noise (misspellings, typos, OCR errors and morphological variations) to the final extraction process. The trade-off between the morphological ambiguity and the resilience originated by the approximate string comparison is going to be studied in this chapter.

Two elements are going to be used in order to deal with the two mentioned sources of ambiguity: *i)* the context in which the ambiguous term is used and *ii)* the information contained in the laptop ontology. However, unfortunately the very laptop ontology adds a third source of ambiguity that is the *use ambiguity*. The use ambiguity is originated by the fact that the ontology is designed as a

directed acyclic graph (DAG) instead of being a tree. Nevertheless, this property in ontology design allows that a concept can be reused. For instance, if the term "MB" (linked to the concept *MegaByte*) is found in a document, it is not possible (without considering additional information) to determine if it is referring to the memory units of the laptop main memory or the cache memory. This ambiguity had briefly outlined in section 2.4 when the possibles semantic paths where considered for labeling a sample of a document sequence. The figure 2.5 showed that the 58.47% of the terminal nodes in the laptop ontology have this kind of ambiguity.

As a consequence of the previously discussed sources of ambiguity, the token sequences in the document corpus can have many semantic interpretations. We noticed that this problem is similar to the *word sense disambiguation* (WSD) problem in the *natural language processing* (NLP) field. Besides, a particular set of approaches to address WSD in natural language text uses a general lexicalized ontology called WordNet [62]. WordNet is structurally similar to our laptop ontology and lexicon. Thus, we proposed a solution to our information extraction problem similar to the WSD solutions that uses WordNet as disambiguation resource.

This chapter is organized as follows. In section 4.1 different approaches to the problem of provide comparison measure between concepts in ontologies are reviewed. The relevant WSD approaches for our disambiguation problem are reviewed in section 4.2. The proposed disambiguation method for our specific information extraction problem is presented in section 5. Additionally, this section presents the final information extractor integrated with the approximate string comparison techniques studied in chapter 3. The experimental evaluation of the proposed system using the laptop ontology, lexicon and corpus is presented in section 5.5. In addition to the experiments carried out with the laptop lexicon, new synthetical lexicons were generated adding noise to the original lexicon in order to evaluate the robustness of the system. Finally, in section 5.6 some concluding remarks are made.

4.1 Semantic Relatedness

Semantic relatedness (SR) is a general level of association between two concepts [72]. For instance, there is an intuitive notion that *VideoAdapter* and *Display* are semantically closer compared with the relatedness between *Battery* and *Speakers* concepts. The SR measures between concepts are the base of many WSD algorithms because they provide information to evaluate the coherence of a discourse.

Semantic similarity is a less general concept compared with SR, because it is defined as a measure for semantic networks connected only with a single type of relationships (usually "is-a" hierarchies). Due to the fact, that the laptop ontology was built mainly with one type of relationships, we do not make distinctions between the SR and semantic similarity measures.

4.1.1 Graph-based Semantic Relatedness Measures

The simplest SR measure in ontology graphs (or semantic networks) is *path length* [77], i.e. the number of edges in the shortest path between two concepts (see figure 4.1).

However, the path length metric can generate misleading results because of in practice, pairs of concepts closer to the ontology root tend to be less related than entities closer to leaves. In order to lessen this situation, Leacock and Chodorow [51] proposed the equation 4.1 where D is the total ontology graph height.

$$SIM_{Leacock\&Chodorow}(a,b) = -\log\left(\frac{pathLength(a,b)}{2D}\right) \tag{4.1}$$

Another approach to the same problem has been proposed by Wu and Palmer [93] using the concept of *lowest common subsumer* (LCS), i.e. the more distant concept from the root, common to two concepts. LCS is also known as *lowest common ancestor*, *lowest super-ordinate* and *most informative subsumer*. For instance, in figure 4.1 *Processor* is the LCS of *Brand* and *GHz*. The Wu&Palmer measure is given the equation 4.2, where the function $depth(x)$ represents the number of edges from the concept x to the root concept.

$$SIM_{Wu\&Palmer}(a,b) = \frac{2.depth(lcs(a,b))}{pathLength(a,b) + 2.depth(lcs(a,b))} \tag{4.2}$$

The path length measure can be written using the LCS and depth(x) functions with the following expression:

$$SIM_{pathLenght}(a,b) = depth(a) + depth(b) - 2.lcs(a,b)$$

Recently Altinas et al. [3] proposed a SR measure combining the "abstractness" of the LCS concept and "concreteness" of the concepts being compared. Alintas et al. measure reached the highest F-measure (i.e. harmonic mean between precision and recall) compared with previously presented metrics in the noun word sense disambiguation task of Senseval-2[1].

Altinas *et al. [3]* proposed a measure combining the concepts of abstractness and concreteness using the following expressions:

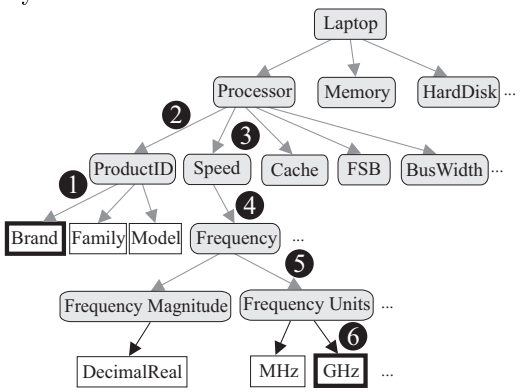
$$LenFactor(a,b) = \frac{pathLength(a,b)}{2.D}$$

$$Spec(x) = \frac{depth(x)}{clusterDepth(x)}$$

$$SpecFactor(a,b) = |Spec(a) - Spec(b)|$$

[1] http://www.senseval.org/

Figure 4.1: Path length semantic relatedness example in a "has-part"/"has-attribute" hierarchy .



$$SIM_{Altinas}(a,b) = \frac{1}{1 + LenFactor(a,b) + SpecFactor(a,b)}$$

The function $clusterDepth(x)$ returns the depth of the deepest node in a cluster of WordNet concepts.

Another approach was proposed by Stetina et al. [84]a semantic distance is defined in the WordNet hypernymy hierarchy using the following expression:

$$sd(a,b) = \frac{1}{2} \times \left(\frac{depth(a) - depth(lcs(a,b))}{depth(a)} + \frac{depth(b) - depth(lcs(a,b))}{depth(b)} \right)$$

Further, the semantic distance is converted in a semantic similarity using the equation 4.3. The similarity is squared in the following formula in order to report higher values to semantically closer terms.

$$SIM_{Stetina}(a,b) = 1 - sd(a,b)^2 \tag{4.3}$$

Sussna [85] proposed a semantic relatedness measure using all types of relationships in the WordNet [62]semantic network (i.e. synonymy, hypernymy, hyponymy, holonymy, meronymy and antonymy). The Sussna's measure between two concepts consist in the shortest path using weighed edges according to a *type-specific fanout* (TSF) factor scaled with the depth of the deeper concept in the hierarchy. The TSF factor reflects the strength of the relation between two adjacent nodes in the semantic network related with the number of like relations that the source node has. For instance the weight of the edge in the meronymy relation between *Processor* and *Laptop* is inverse to the number of meronymy relations between *Laptop* and other concepts. Similarly, the weight for inverse relation (i.e. holonymy between *Laptop* and *Processor*) is obtained in the same way. Both values are averaged and scaled obtaining the undirected edge weight as is shown in the following expression.

$$weight(a,b) = \frac{w(a \rightarrow_r b) + w(b \rightarrow_{r'} a)}{2d}$$

$$w(x \rightarrow_r y) = max_r - \frac{max_r - min_r}{n_r(x)}$$

Where a and b are concepts in the hierarchy,$\rightarrow_r$is a relation of type r, $\rightarrow_{r'}$ is its inverse, d is the depth of the deeper of the two nodes, $max_r$and $min_r$are the range of possible weights for a relation of type r, and $n_r(X)$ is the number of relations of type r having x as source node. Sussna assigned $max_r = 2$ and $min_r = 1$ for all relation types except antonymy whose range was $(2.5, 2.5)$ i.e. no range.

Agirre and Rigau [2] proposed a conceptual distance using not only the path length between concepts and their depth in the hierarchy, but adding the density of terms in the hierarchy as evidence. They argue that concepts in a dense part of the hierarchy are relatively closer than those in a more sparse region.

Maynard and Ananiadou [56] proposed another similarity measure and applied it in the biomedical UMLS[2] ontology.

$$SIM_{Maynard\&Ananiadou}(a,b) = \frac{2.depth(lcs(a,b))}{depth(a) + depth(b)}$$

4.1.2 Corpus-based Semantic Relatedness Measures

Another SR measures uses the concept of *information content* (IC) introduced by Resnik [78].

$$IC(c) = -log(P(c)) \tag{4.4}$$

In equation 4.4 $P(c)$ is the probability of encountering an instance of the concept c in some specific and large corpus. Using the IC concept is possible to bring together the information of an ontology and a large corpus. The former is considered a *knowledge-rich* source and the later a *knowledge-poor* source. Budanitsky and Hirst [15] made a comparative study showing that semantic relatedness measures that combines those sources have better results in WSD task when a large corpus is available. The SR measure proposed by Resnik uses the following expression:

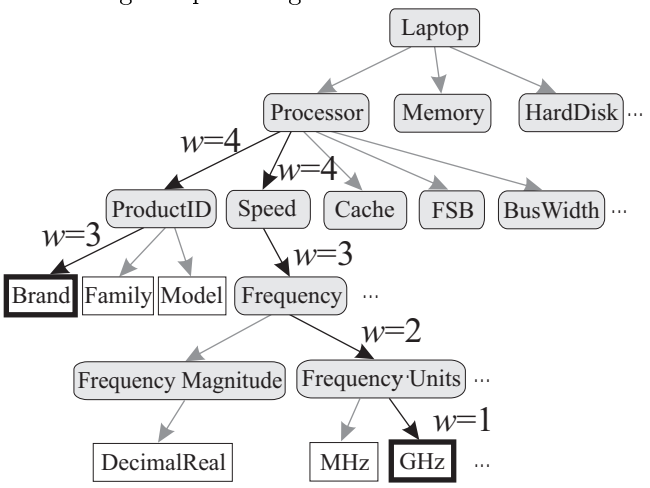
$$SIM_{Resnik}(a,b) = IC(lcs(a,b))$$

Using the same IC concept, Jiang and Conrath [44], proposed the following measure:

$$SIM_{Jiang\&Conrath}(a,b) = IC(a) + IC(b) - 2 * IC(lcs(a,b))$$

[2] http://www.nlm.nih.gov/research/umls/

Figure 4.2: Weighted path length semantic relatedness measure example.



WeightedPathLength("Brand","GHz")=3+4+4+3+2+1=17

Similarly, Lin [54]proposed the following SR metric:

$$SIM_{Lin}(a,b) = \frac{2 * IC(lcs(a,b))}{IC(a) + IC(b)}$$

4.1.3 A New Semantic Relatedness Measure

Leacock et. al and Wu & Palmer measures have extended the basic path length metric (see figure 4.1) following the idea that two concepts closer to the ontology hierarchy root are semantically more distant than those that are farther. Using that idea, we propose a weighted path length measure assigning weights to edges according to its proximity to the root of the ontology tree. In figure 4.2 an example of that metric is shown.

4.2 Knowledge-Based Word Sense Disambiguation

Word sense disambiguation (WSD) is a natural language processing task that aims to assign the correct sense to all polysemous words in a text [60]. Firstly, in order to define the search space for the WSD problem it is necessary an inventory of possible senses for each word in a text. That source of senses is usually a machine readable dictionary [52] or a semantic network such as WordNet.

The following example has each open class word sub scripted with the number of senses found in the on-line version of The American Heritage Dictionary

of the English Language[3] .

"Jackson found(4) good(42) medicine(9) in his summer(7) field(37) trips(23) with their unfolding(6) of new(15) horizons(8)"[4]

The number of possible sense combinations in that sentence is 6,485'028,480. SR metrics are used to assess the semantic coherence of a specific sense combination. The WSD goal is to find a sense combination that maintains logical discourse coherence. Due to high dimensionality and huge search space of the problem, techniques such as simulated annealing [26] and genetic algorithms [39] have been used to find near-optimal combinations.

4.2.1 Gloss overlap WSD Algorithms

Gloss overlaps, relies on the idea that gloss overlaps (i.e. words in common between the dictionary definitions of a targeted word and its surrounding words (i.e. context) or glosses from that surrounding words are some kind of SR measure. Lesk [52]first proposed this approach using glosses from a *machine readable dictionary* (MRD). The sense combination whose glosses have the highest overlap (i.e. words in common) is assumed to be the correct one. Another approach known as *simplified Lesk* proposes to disambiguate a targeted word searching the highest gloss overlap between the glosses of the targeted word with the context words. The simplified Lesk algorithm do not use the previously disambiguated word to disambiguate the next. This simplification reduces considerably the search space and surprisedly get better accuracy than the original approach [47].

Extended Gloss Overlaps [8] uses not only the overlaps with neighboring words glosses, but extend the comparisons to glosses of related words. Using this additional information the disambiguation process is improved.

Cowie *et al.* [26] proposed a global coherence WSD method at sentence level. The method can be seen as an application of the Lesk's algorithm simultaneously to all ambiguous words in a text. The coherence evaluation of a candidate sense combination is made using the set of sense glosses. Redundancy factor R is computed in the set of glosses by giving a stemmed word from which appears n times a score of $n-1$ and adding up the scores. Finally the sense combination with the highest R factor in its glosses is selected using simulated annealing as optimization technique.

Gelbukh *et al.* [39] proposed a global coherence method at document level using a semantic relatedness measure based in glosses. The method aimed to find an optimal sense combination at document level that minimizes the average distance between senses. Due to the great search space, a genetic algorithm was used to find a near optimal solution. The fitness function is similar to the mutual constraint principle defined by Sussna [85]. Experiments showed that

[3] `http://www.thefreedictionary.com`

[4] Findley Rowe. The Life and Times of William Henry Jackson Photographing the Frontier. National Geographic Magazine, vol 175, No. 2, 1989 p.238.

global coherence hypothesis gave better results than methods that uses only local context such as Lesk's algorithm.

4.2.2 Semantic-Relatedness-based WSD Algorithms

Banerjee and Pedersen [7, 8] noticed that the Lesk's algorithm can be expressed as an algorithm based in a semantic relatedness measure and name it the *Adapted Lesk Algorithm*. In the same way as the simplified Lesk's algorithm, a targeted word is disambiguated using a symmetric context window of size $2n + 1$. Let $s_{i,j}$be the i-th sense of the word at the j position relative to the target word $j = 0$. The SenseScore for sense $s_{0,k}$is computed as follows:

$$SenseScore_k = \sum_{i=-n}^{n} \sum_{j=1}^{|w_i|} relatedness(s_{0,k}, s_{i,j}),\ i \neq 0$$

Where $|w_i|$ is the number of senses of the word at the relative position i. The algorithm chooses the sense with the highest $SenseScore_k$ for the targeted word.

Sussna [85] proposed a strategy called *mutual constraint*, which consist of the addition of all pairwise semantic relatedness measures in a sense combination of a text. The candidate sense combination that maximizes the mutual constraint gives the most coherent sense combination. This approach is only practical at sentence level or within a context window due to the great number of possible sense combinations. This approach also agrees with the one-sense-per-discourse hypothesis because different senses in two instances of a single word in one text convey a sub-optimal mutual constraint.

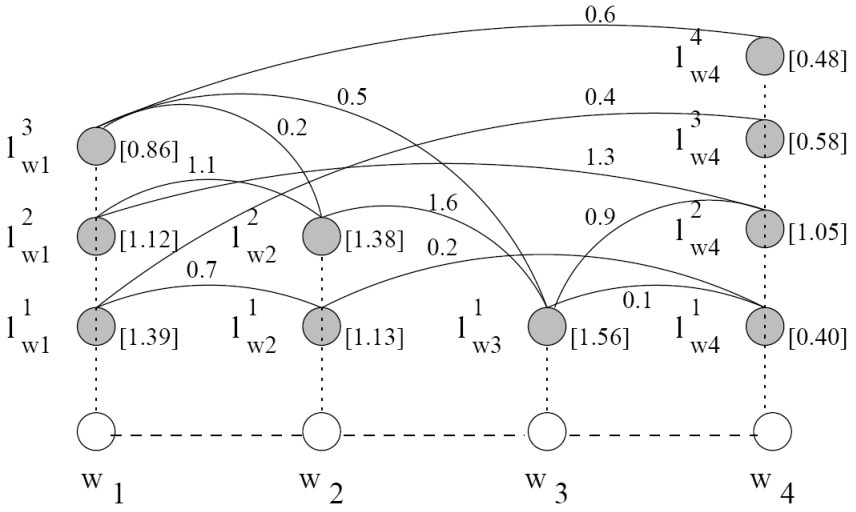
Mavroeidis et al. proposed a method called *compactness based disambiguation* [55], used initially for document classification. The possible sense combinations are treated as a set and the most coherent combination is considered. This combination is obtained computing the Steiner tree [43] in the semantic network formed with all the possible word senses in the discourse. The Steiner tree is restricted to include paths through all least common subsumers (LCS) of the set of senses.

4.2.3 Graph-based WSD Algorithms

Mihalcea et al. [61] proposed the use of PageRank algorithm [14] in a graph built with nodes representing word senses in an text and edges are semantic relations extracted from WordNet. Then PageRank is used to assess the "popularity" of each node (*i.e.* sense) in a directed graph representing web pages as nodes and edges as hyperlinks. The idea of PageRank can be extended to undirected graphs, in which the out-degree of a node is the same in-degree. Let $G = (V, E)$ be a undirected graph with the set of nodes V and a set of edges E. The PageRank value for a node V_i is defined with the following expression:

$$PageRank(V_i) = (1 - d) + d * \sum_{j \in neighbors(V_i)} \frac{PageRank(V_j)}{|neighbors(V_j)|}$$

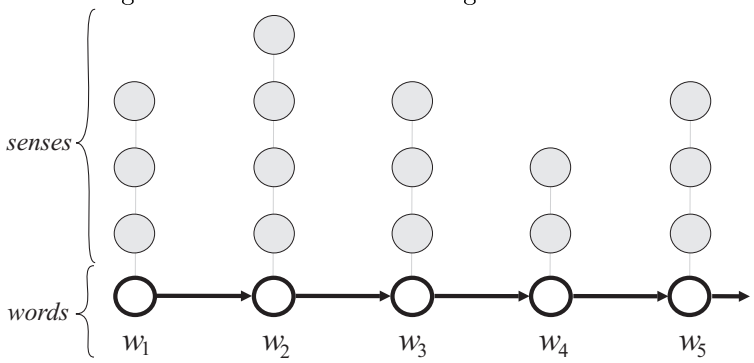
Figure 4.3: Sample graph built over the set of possible labels (shaded nodes) for a sequence of four words (unshaded nodes). Label dependencies are indicated as edge weights. Scores computed by the graph based algorithm are shown in brackets, next to each label (taken from Sinha and Mihalcea [59, 83])



Where the function $neighbors(V_i)$ denotes the set of adjacent nodes to the node V_i, and d is a damping factor set in 0.85 (see [14] for details). Scores for each node are initialized in 1 and PageRank is calculated iteratively until convergence. Finally, for each word, the sense with the highest PageRank is selected. This method is in agreement with the one-sense-per-discourse hypothesis [38] because of nodes representing the same sense obtains the same PageRank score. The authors reported better results for their method (47.27% average accuracy) compared with the Lesk's algorithm (43.19% average accuracy).

Sinha and Mihalcea[59, 83] proposed a method for WSD constructing a graph aligned to the document whose nodes are possible senses for each word and edges are semantic similarity measures. Edges are only allowed within a sliding window over word sequence (size 6 in the experiments) and above a similarity threshold. Further, a *graph-based centrality measure* [67] (i.e. indegree, closeness, betweenness and PageRank) is computed for each node to determine a weight. Nodes (i.e. senses) with higher weights are selected for disambiguation. Figure 4.3 shows a sample graph.

Figure 4.4: Word sense disambiguation scenario.



4.3 A New Graph-Based Term Disambiguation Strategy: Shortest-Path

The WSD algorithms reviewed in the previous section considered the disambiguation procedure at word level. This means, that each word in the text has a set of possible senses aligned in the position of the word in the document. The sense inventory is provided by a machine readable dictionary or a lexicalized ontology such as WordNet. The figure 4.4 depicts this scenario. This picture is clearly similar to the graph in figure 4.3.

As it was mentioned in the introduction of this chapter, our intention is to propose a term disambiguation strategy inspired in the WSD algorithms. Nevertheless, the terms unlike the words are compound of one or more tokens (words). The case of terms with more than one token is very common in the laptop lexicon as it can be seen in figure 2.9. Additionally, the sense inventory is generated by the three sources of ambiguity also mentioned in the introduction of this chapter.

Polysemy: terms associated to many concepts.

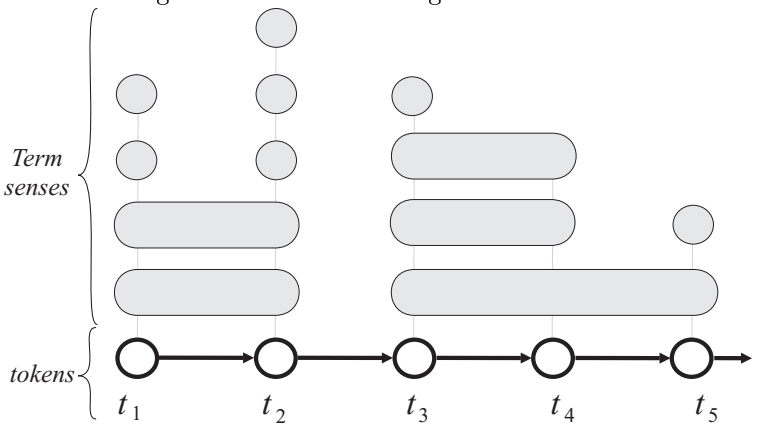
Morphological_similarity: terms whose character strings representations are similar.

Terminal_concepts_use_ambiguity: terms associated to concepts that are shared in the ontology.

The possibility of terms with many tokens and the sources of ambiguity generate a disambiguation scenario with a shape different to the traditional WSD scenario showed in figure 4.4. The candidate senses can range many tokens and a token can have senses ranging different number of tokens. The term disambiguation scenario has a graphical shape that can be seen in figure 4.5.

The sense combination in the term disambiguation scenario has to to guarantee that any token be covered by only one sense. For instance, in figure 4.5

Figure 4.5: Term disambiguation scenario.



the sense that ranges from t_3 to t_5 can not be selected simultaneously with the second sense for the token t_5. Despite the difference of the scenarios between WSD and term disambiguation, most of the WSD algorithms can be adapted to term disambiguation.

Another problem that arises when the WSD algorithms are being adapted to term disambiguation is that the one-sense-per-discourse hypothesis can not be accepted. The reason is due to the fact that, if in the possible senses of a term are included also their possible uses, then only use per discourse can be accepted for a term. For instance, it can not be accepted that all occurrences of the term 'MB' in a discourse be related only to the units of the cache memory. Consequently, the WSD algorithms that are in agreement with that hypothesis do not seem to be suitable to the proposed disambiguation task.

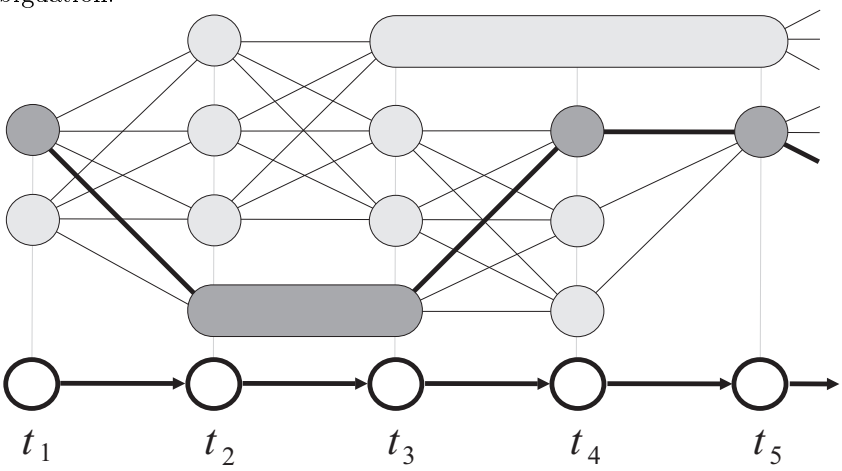
For our specific disambiguation task, a criteria that have not been used in the past is proposed. The approach is graph-based (see section 4.2.3). The idea is to build a graph aligned to the document token sequence where the nodes are the term senses, the undirected edges are added between adjacent senses and the edges are weighted with the semantic distance (i.e. the inverse of the semantic relatedness) between the senses. An outline of this graph is shown in figure 4.6.

We argue that the shortest path in this graph is a coherent disambiguation strategy. This idea is motivated by the fact that consecutive terms in a text document tends to be semantically close. So, in a coherent discourse the chain of terms are connected with small semantic distances. The combination that minimizes the addition of those semantic distances is the shortest path in the proposed graph.

This reasoning could not be valid in texts written in natural language due to phenomena like anaphoras and long term relations. However, the data-sheet documents that we are considering are free of this type of issues.

The shortest path in the proposed graph can be obtained using the Dijkstra's

Figure 4.6: A sample of the graph aligned to the token sequence for term disambiguation.



algorithm [30] of with the Viterbi's algorithm [89] due to the Markovian property of the graph. For our implementation we chose the Dijkstra's algorithm.

Chapter 5

An Information Extraction System for Data-Rich Documents

This chapter assembles the the building blocks developed in the three last chapters in an information extraction system. A block diagram with the overall system architecture is shown in figure 5.1. The architecture is going to be outlined describing each one of the numbered arrows in that figure.

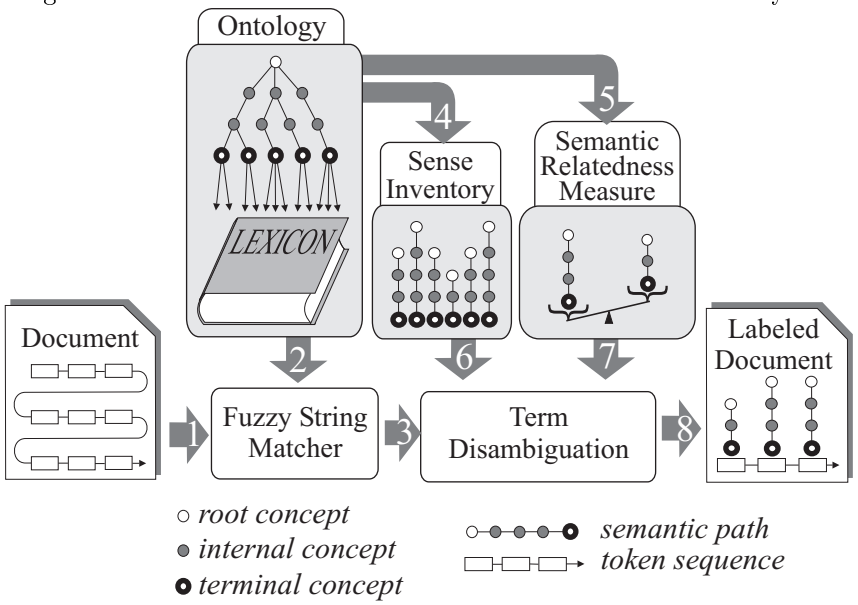
1. The input to the system is a sequence of tokens obtained from the text of a laptop data-sheet (see section 2.4) . The character sequences were tokenized with a rule-based procedure described in section 6.2.1.

2. The list of terms of the laptop lexicon (see section 2.3) and the document sequences are provided as input to the Fuzzy String Matcher. This module make use of the techniques studied in chapter 3 for searching occurrences of the lexicon terms in the documents in a resilient way.

3. The set of occurrences found by the Fuzzy String Matcher is the main input for the term disambiguator. Each occurrence stores the positions of the document sub sequence that were matched and its associated concept in the ontology (usually a terminal concept). It is important to note that the tokens in the sequence can have several valid matches. In addition, this matches can range several tokens in the sequence and also match overlaps are allowed.

4. The ontology is a directed acyclic graph (see section 2.2), that means that concepts can be referenced several times. The possible uses of a specific concept can be obtained enumerating the possible paths from the root concept to that concept. This enumeration constitutes the ontology sense inventory of semantic paths.

Figure 5.1: Overall architecture for the Information Extraction system



5. The implemented semantic relatedness measures were computed using only the ontology graph of the ontology graph.

6. For each matched term in the document a set of senses or uses are provided by the sense inventory. The term disambiguator replaces each matched term label with a set of possible semantic interpretations (i.e. a set of semantic paths).

7. A mechanism to compare two semantic interpretations (i.e. semantic paths) is provided to the term disambiguation module in order to assess the coherence of a specific sense combination over the document.

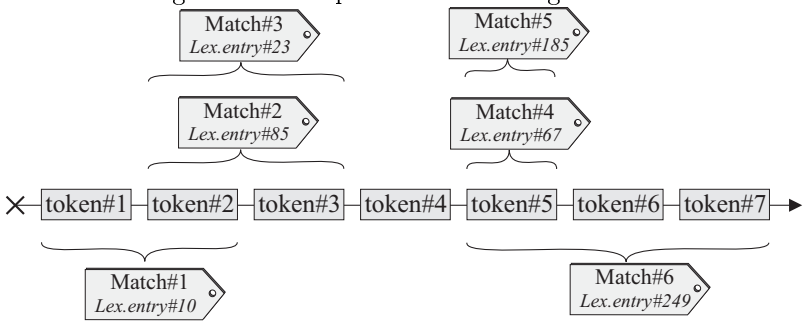
8. Finally. the term disambiguation module combines the inputs 3., 6. and 7. to decide an unique semantic interpretation to each token in the original document token sequence.

The sections 5.1, 5.2, 5.3 and 5.4 give additional details about the implementation of the different modules of the system. The experimental validation of the extraction system is provided in section 5.5. Finally, some conclusion remarks are made in section 5.6.

5.1 The Fuzzy String Searcher

The *fuzzy string matcher* (FSM) module searches for fuzzy coincidences of the laptop lexicon into the laptop data-sheets documents. The terms in the laptop

Figure 5.2: Sample of the matching schema.



lexicon can have one or more tokens, so each match label over the document stores three fields: the lexicon entry name and the position of the starting and ending token. The diagram in figure 5.2 outlines the matching labeling schema. There are some additional issues related to the matching that are listed next:

- Overlaps between matches are allowed.
- If two terms in the same lexicon entry match the same tokens only one match is reported.
- If two consecutive matches refers to the same lexicon entry both matches are not replaced by a single match.

The naïve approach to match the entire lexicon against the document is to check all token positions with all lexicon terms. However, this approach is prohibited because the number of lexicon terms is greater than 1500 and the evaluation corpus has more than 5000 tokens. The approaches to solve this problem such as suffix trees, automatas are out of the scope of this thesis.

All the approximate string comparison techniques studied in chapter 3 were available in the FSM. Nevertheless, we implemented a simple blocking strategy with a term index by the first bi-gram of characters. Clearly, all the tokens (nor only the first) of the terms is included in the index. Additionally, all terms that seem to be acronyms (i.e. only with capital letters and with less than 7 characters) are indexed by only by their first letter. For instance, the "USB" term was included in all the bi-gram index entries whose first character is "U". This blocking approach is very conservative, but in preliminary experiments showed to be slow but very effective.

The pattern obtained from the lexicon that is being compared against the document has considerably less tokens than the later. So, it is assumed that in all cases (except at the end of the document) the document provide enough tokens to be compared with the pattern. However, if a punctuation mark is present in the document it is reasonable to think that tokens beyond this punctuation mark should be ignored. For instance, if the pattern "Hard Disk Drive" is being compared in in the following document fragment "... hard disk. Optical drive:

Table 5.1: List of boundary separators

Character	Name
.	period
,	coma
;	semicolon
:	colon
(	open parentheses
)	close parentheses
[	open brackets
]	close brackets
?	question mark
¿	open question mark
!	exclamation mark
¡	open exclamation mark
<CR>,<LF>	carriage return, line feed

DVD- ...", the period after the token "disk" in the document is boundary and the token "Drive" from the patten should not be compared with "Optical". Those boundary separators are listed in table 5.1.

5.2 The Use/Sense Inventory

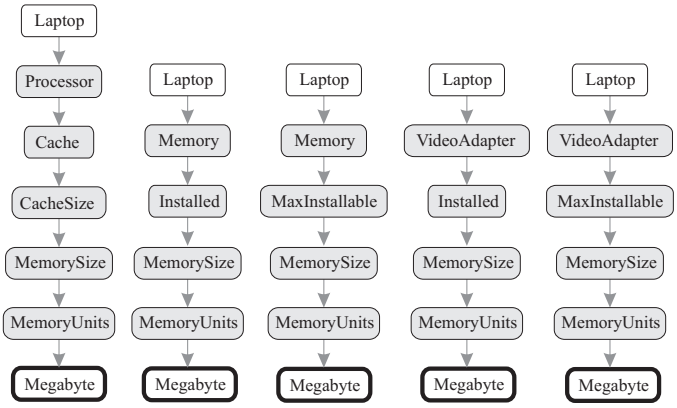
The use/sense inventory is a simple repository that stores all the possible semantic paths for each terminal node in the ontology (see the semantic path definition in section 2.2). This inventory is implemented with a dictionary whose key is a terminal concept name and the data is a set of semantic paths. This inventory is preprocessed in order to improve the requests of the term disambiguation module.

5.3 The Implemented Semantic Relatedness Measures

Five semantic relatedness measures among the measures reviewed in section 4.1 were implemented:

1. Path Length [77] (baseline).
2. Weighted Path Length (see section 4.1.3)
3. Wu & Palmer measure [93].
4. Maynard & Ananiadou measure [56].
5. Stetina et al. measure [84]

Figure 5.3: A sample of semantic paths for the terminal concept *MegaByte*.



The selected measures were standardized as semantic distances instead of semantic similarities. The measures 1. and 2. are already distances. Due to the fact that the values of measures 3. and 4. were normalized in the range [0,1], they were converted as distances with the simple expression $distance = 1 - normalizedSimilarity$. The Stetina et al. similarity measure were converted as a distance using the equation 4.3 proposed by the author.

The semantic relatedness measures that uses large corpus (see section 4.1.2) were not used because in the laptop domain it is difficult to gather a really large document corpus of data-sheets in order to reach enough statistical evidence for each target. On the other hand, the string similarity measures studied in chapter 3 were only static measures, therefore they either do not use additional information of the corpus. So, this selection were also made in order to be consistent with the used string matching.

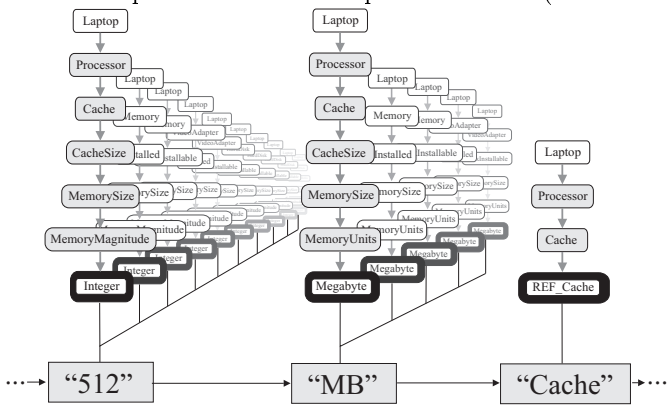
5.4 The Term Disambiguation Module

The term disambiguation module aims to select one unambiguous semantic interpretation to each one of the tokens in the input document. This module receives as input the output of the fuzzy string matcher (see figure 5.2) and uses the sense inventory in combination with a semantic relatedness measure in order to select a coherent semantic interpretation of the text.

Three disambiguation strategies were implemented and two baselines:

1. Shortest path, see section 4.3.
2. Adapted Lesk Algorithm [71] with a window of 3 tokens.
3. Shortest path but evaluated only in a sliding window o 3 tokens.
4. First sense selection (baseline).

Figure 5.4: Token sequence labeled with possible senses (*i.e.* semantic paths)



5. Random sense selection (baseline).

The input from the fuzzy string matcher firstly is combined with the use/sense inventory. As a result, the document token sequence get labeled with the possible semantic paths for each identified term. An example of a token sequence sample at this point is shown in figure 5.4. Further, each semantic path is considered as a node in a kind of view from above of figure 5.4 in figure 5.5. It is important to note that this sample ("512MB Cache") do not include multi-token terms. So, the shape of the graph is closer to the figure 4.5 in a commoner scenario.

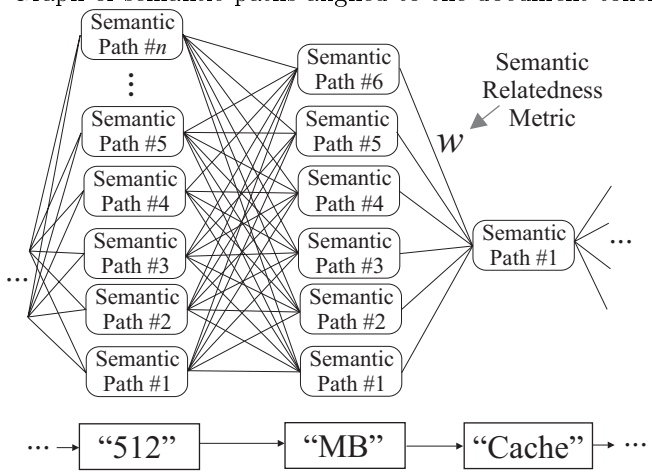
The edges added to the graph in figure 5.5 are weighted with semantic distances obtained from one of the implemented semantic relatedness measures. Finally, one of the implemented disambiguation strategies select an unique semantic path for each token which had candidates. The final output is the same input sequence of tokens with selected tokens labeled with a semantic path.

5.5 Experimental Validation

The experiments were designed to answer the following questions:

- Which are the static fuzzy string matching techniques more suitable to the proposed information extraction task?
- Do fuzzy string matching techniques provide robustness or noise to the proposed information extraction task?
- Which of the selected and proposed semantic relatedness measures makes better the term disambiguation process?
- Which of the selected and proposed disambiguation strategies are the more convenient in the specific information extraction task?

Figure 5.5: Graph of semantic paths aligned to the document token sequence



The experiments were carried out using the manually labeled corpus described in section 2.4 combining each one of the following variables:

1. The approximate string matching technique.
2. The semantic relatedness measure.
3. The disambiguation algorithm.
4. The lexicon. The options were the original corpus and 8 synthetically generated noisy lexicons.

The synthetically generated noisy lexicons were obtained making at random edit operations in the original laptop lexicon. The level of noise P was established in order to control the amount of noise inserted in the original lexicon. Four levels of noise were established: 25, 50, 75 and 100. The noise level $P = 0$ means the original noise-free lexicon and 100 or more the maximum noise level. The procedure to generate the noisy lexicons is presented in algorithm 3.

The four generated lexicons with random edition operations at character level are listed in table 5.2 with the number or edited tokens and the percentage of edited tokens of the 2771 tokens in the lexicon.

The other set of noisy lexicons were generated with the same procedure described in algorithm 3, but additionally shuffling the tokens in the lexicon terms. This procedure is coarsely described in algorithm 3.

In the same way, the four additional generated lexicons are described in table 5.3 (the total number of terms is 1700).

Algorithm 2 Noisy lexicon generator with edition operations at character level.

```
function noisyCharacterEdition(LEXICON, P)
    for all TERM in LEXICON do
        for all TOKEN in TERM do
            if length(TOKEN)>2 and
               randomIntegerInRange(0,100)<P then
                position=randomIntegerInRange(2,length(TOKEN))
                operation=randomIntegerInRange(1,3)
                if operation=1 then
                    TOKEN=deleteCharacterAtPosition(position,TOKEN)
                if operation=2 then
                    TOKEN=insertCharacterAtPosition(position,TOKEN)
                if operation=3 then
                    TOKEN=replaceCharacterAtPosition(position,TOKEN)
    return(LEXICON)
```

Table 5.2: Description of the noisy generated lexicons with character edition operations.

P	# edited tokens	edition rate
25	510	18.40%
50	1034	37.31%
75	1553	56.04%
100	2109	76.10%

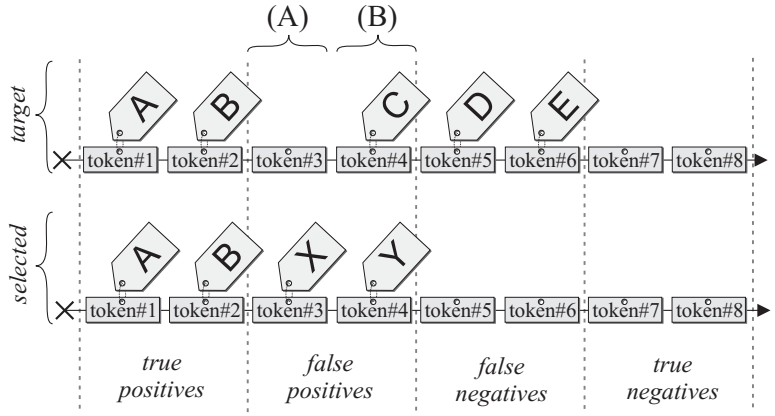
Algorithm 3 Noisy lexicon generator shuffling the tokens of the terms.

```
function shuffledTokenEdition(LEXICON, P)
    LEXICON=noisyCharacterEdition(LEXICON, P)
    for all TERM in LEXICON do
        if numberOfTokensIn(TERM)>1 and
           randomIntegerInRange(0,100)<P then
            TERM1=shuffleTokensIn(TERM)
            while TERM1=TERM do
                TERM1=shuffleTokensIn(TERM)
            TERM=TERM1
    return(LEXICON)
```

Table 5.3: Description of the noisy generated lexicons with term shuffling at token level.

P	# edited tokens	edition rate	# shuffled terms	shuffling rate
25	547	19.74%	191	11.23%
50	1060	38.25%	391	23.00%
75	1575	56.83%	628	36.94%
100	2109	76.10%	857	50.41%

Figure 5.6: Characterization of the labeling problem as a two class problem.



5.5.1 The Performance Metric

The problems of labeling a sequence with a set of labels can be evaluated in may ways. In fact, in a problem with n labels there are $(n+1)^2 - n$ type of errors. That is, a confusion matrix of dimension $(n+1) \times (n+1)$ (the additional label is the "no-label" label) withdrawing its diagonal. However, in order to make an analysis of those errors it is necessary to have a very large corpus if the number of possible labels is large too. That is our case.

Another option, is to considering the labeling as a two class problem. That is, a particular target token has been labeled or not and the selection was correct or not. The picture in figure 5.6 shows that approach. The *target* labeling is the manually labeled laptop evaluation corpus (section 2.4) and the *selected* labeling is the output of the information extraction system. Each token in the selected labeling sequence falls in one the next four categories:

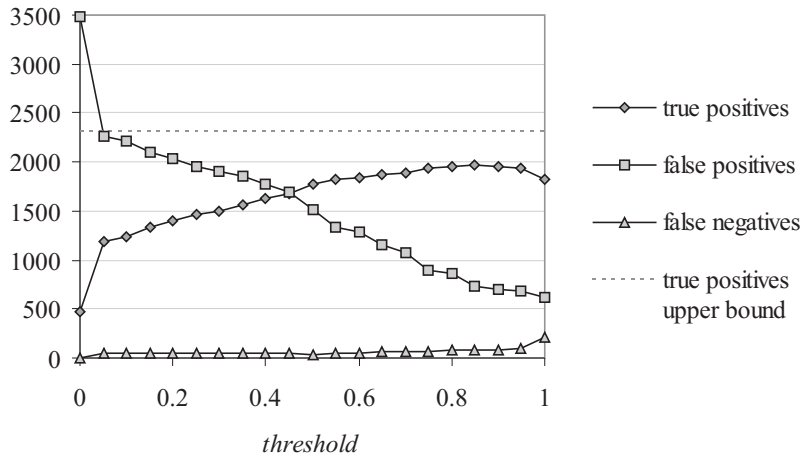
True Positive (TP): When the token has been labeled and the selected label is the same target label. This is the result of an accurate string matching and/or an accurate term disambiguation.

False Positive (FP): When the token has been labeled and the selected label is incorrect. There are two cases in this category: (A) when the target node had no label and (B) when the target node had a different label. The case (A) is the result of a poor string matching and the case (B) is the result also of a poor string matching and/or an in-accurate term disambiguation.

False Negative (FN): When the token has not been labeled even though, the target token had a label. This category is the result of a poor string matching.

True_Negative (TN): When the token has not been labeled and the target token had no label either.

Figure 5.7: Counting of true positives, false positives and false negatives ranging the string matching threshold in (0,1]. This result was obtained using edit distance at character level, Minkowski distance at token level ($p = -10$), path length as semantic relatedness measure, shortest path as disambiguation strategy using the noise-free laptop lexicon.



The graphic in figure 5.7 shows the tendency of TP, FP and FN when the string matching threshold is ranged form 0 to 1. The threshold value of 0 means that all string comparisons among the strings compared in the set defined by the blocking strategy are considered as valid matches. The threshold value of 1 means that only identical strings are considered as valid matches. It is important to note that the counting of FN is low because of the used lexicon is noise-free. However, if a noise lexicon were used, the FN plot would have a clearly increasing tendency.

The counting of TP, FP and FN is used to define the *precision*, *recall* and *F-measure* measures with the following equations:

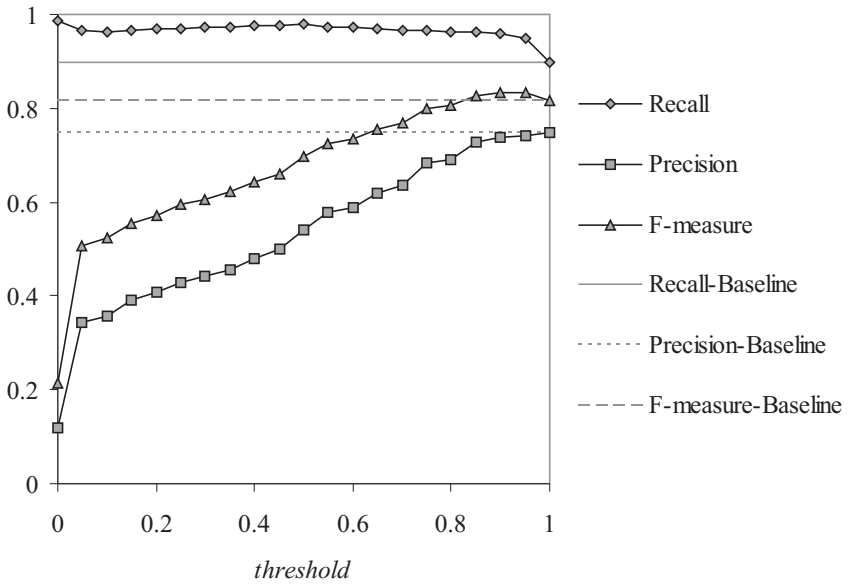
$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F - measure = \frac{2 \times precision \times recall}{precision + recall}$$

The precision measure can be interpreted as the rate of correct labeled tokens in the set of selected labeled tokens. Similarly, the recall measure is the rate of

Figure 5.8: Precision/recall/F-measure vs. threshold plot for the same experiment of figure 5.7. Baselines are obtained with the same experimental setup but changing the string matching technique to exact match.

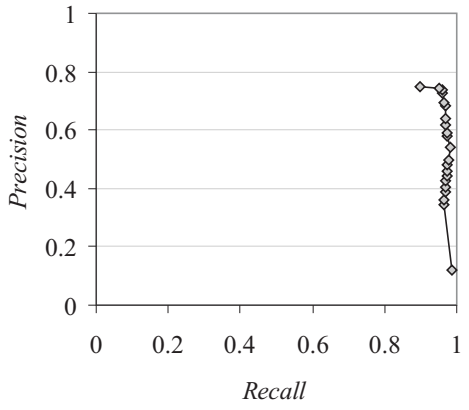


correct labeled tokens in the set of target labeled tokens. The F-measure is the harmonic mean between precision and recall and reflects the balance of the two measures when both have high values. The graphic in figure 5.8 shows a plot of precision, recall and F-measure for the same experiment of figure 5.7.

The performance metric used for the string matching problem in chapter 3 was the interpolated average precision (IAP). This metric was obtained interpolating the recall-precision curve for different thresholds and the area under that curve is the IAP metric. The graphic in figure 5.9 shows the recall-precision plot for the same experiment of figure 5.7. The interpolated would look like a step with its elbow in the point where precision and recall have their combined higher value. This point is also the maximum value of F-measure and is called as *F1-score*. The F1-score is a measure of general performance of the labeling process but focused in the point where the performance reach its maximum. The string matching threshold were range using steps of 0.05.

The IAP measure reflects the performance of the labeler for the complete range of values of the threshold. However, there is not too much interest in the performance of the labeler for small values of the threshold. Thus, the F1-score seems to be a more convenient quantitative measure to our particular extraction system. Additionally to the F1-score, the plot of precision/recall/F-measure vs. threshold is reported in some experiments as a qualitative measure. Those plots

Figure 5.9: Recall-precision plot the same experiment of figure 5.7.



also gave important information about the string matching thresholds where the F1-score was reached.

5.5.2 Results for the Fuzzy String Matcher

The proposed information extraction system in this chapter is mainly composed of two subsystems: the fuzzy string matcher and the term disambiguation module. The performance of the overall system can be evaluated with the metrics proposed in the previous section. However, an evaluation data set only for the fuzzy string matcher is not available. Thus, in order to evaluate only the fuzzy string matcher the semantic relatedness and the disambiguation strategy were preestablished for all the experiments. Farther, experiments varying the string matching technique were carried out. It is expected that the changes in the overall performance were only originated by the accuracy of the string matching technique. The used semantic relatedness measure was path length (section 4.1) and the disambiguation strategy was shortest path (section 4.3).

It is important to note that the lower the string matching threshold, the higher the number of valid matches and consequently, the higher the ambiguity that the term disambiguation module has to deal with. The graphic in figure 5.10 shows the number o valid matches vs. the string matching threshold using as string matching technique EditDistance-MongeElkan[1].

The experiments were carried out varying five string matching techniques at character level (i.e. EditDistance, 2grams(Dice), Jaro, ExactMatch and JaroWinkler) and five combination techniques at token level (i.e. SimpleStr, Minkowski-10, MongeElkan, MongeElkan0 and cosine(A)). Results using the noise-free laptop lexicon for the F1-score are listed in table 5.4 and the string

[1]The naming convention for the string matching techniques is the same used in chapter 3

Figure 5.10: Number of valid matches for different string comparison thresholds using EditDistance-MongeElkan as matching method.

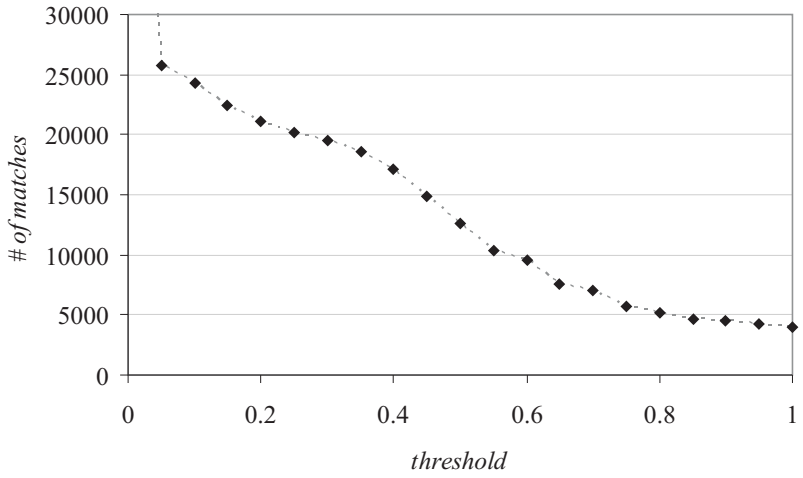


Table 5.4: F1-score for different string matching techniques using the noise-free laptop lexicon.

	EditDistance	2gram(Dice)	Jaro	ExactMatch	JaroWinkler
SimpleStr	0.833443	0.831804	**0.836105**	(0.816071)	0.816479
Minkowski-10	0.831796	0.829362	0.833693	0.812196	0.831784
MongeElkan	0.835548	0.830689	0.831611	0.813544	0.829053
MongeElkan0	0.830683	0.831666	0.831611	0.813544	0.829053
cosine(A)	0.832407	0.831816	0.747484	0.815807	0.741935
AVERAGE	**0.832775**	0.831067	0.816101	0.814233	0.809661

matching thresholds where that F1-score were reached are listed in table 5.5. The best result and the best average result are shown in bold characters.

The F1-score of 0.816071 obtained with the matching technique ExactMatch-SimpleStr (i.e. exact string comparison) can be considered a upper-bound because as it was mentioned in section 2.3, the laptop lexicon was made based partially in the documents of the laptop evaluation corpus. However, several string matching techniques outperformed slightly this upper-bound revealing that even in the five documents of the evaluation corpus there were morphological variations that have not been considered in the lexicon.

The fact that the approximate string matching techniques has outperformed the upper-bound is a weak probe of the convenience of using those techniques instead of the simple exact matching. However, in order to provide a stronger probe experiments with the noisy synthetical lexicon obtained with the algo-

Table 5.5: F1-score for different string matching techniques using the noise-free laptop lexicon.

	EditDistance	2gram(Dice)	Jaro	ExactMatch	JaroWinkler
SimpleStr	0.90	0.85	0.95	-	1.00
Minkowski-10	0.90	0.75	0.95	-	0.95
MongeElkan	0.90	0.85	0.95	0.85	0.95
MongeElkan0	0.85	0.85	0.95	-	0.95
cosine(A)	0.90	0.90	1.00	0.95	1.00

Table 5.6: F1-score results for different string matching techniques at different noise levels in the lexicon.

	ExactMatch	ExactMatch	EditDist.	EditDist.	2grams(Dice)	EditDist.
P	SimpleStr	MongeElkan	SimpleStr	MongeElkan	MongeElkan	cosine(A)
0	0.816071	0.813544	0.833443	**0.835548**	0.831666	0.832407
25	0.636644	0.705415	0.741921	**0.784871**	0.784615	0.778088
50	0.440036	0.537177	0.662466	0.753542	**0.754949**	0.746999
75	0.255391	0.418154	0.587655	**0.716136**	0.710129	0.704862
100	0.081679	0.207174	0.529873	**0.699073**	0.682116	0.683386

rithm 3 and four string matching techniques were carried out. The results of the F1-score for different levels of noise in the lexicon are shown in figure 5.11. As it was expected, the results show that the overall system performance is better maintained when an approximate string comparison technique is used at character level.

The lexicons generated with additional noise at token level (see algorithm 3) were used in a similar way in order to evaluate the resilience of the same matching methods. The results are shown in figure 5.12. This graphic clearly shows that a matching technique with a fuzzy strategy at character and at token level such as EditDistance-MongeElkan provides the better resilience against the noise.

The table 5.6 shows the tabulated results of the curves of figure 5.12 with two additional columns with the results of the matching methods 2grams(Dice) and EditDistance-cosine(A). Characters in bold face shows the better results for each noise level.

There are another important issue related to the use of approximate string matching techniques in the extraction process, that is to determine the right value for the string matching threshold. The figure 5.13 shows a matrix of precision/recall/F-measure vs. threshold plots for 5 levels of noise in the lexicon (generated with the algorithm 3). It can be noticed that the threshold where the F1-score is reached tended to move to the left in the plots when the level of noise in the lexicon increases.

The threshold values where the F1-score were reached are showed in table 5.7. Surprisedly, we notice that the combination method at token level with the

Figure 5.11: F1-score for different matching methods using five levels of character-edition noise in the lexicon.

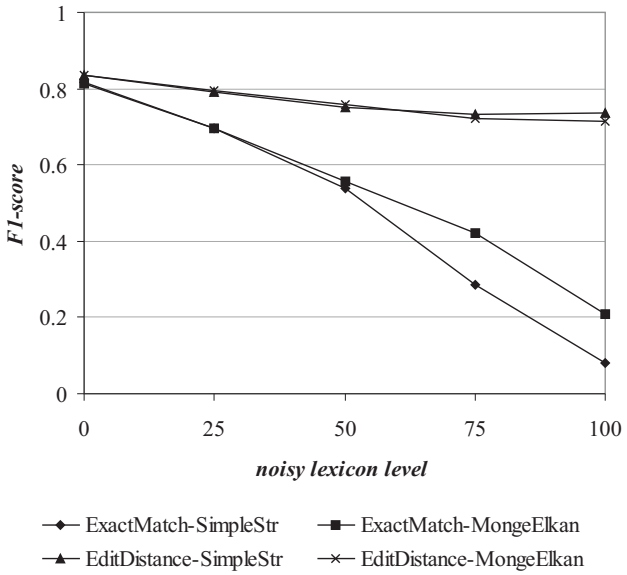


Figure 5.12: F1-score for different matching methods using five levels of character-edition noise combined with token shuffle noise in the lexicon.

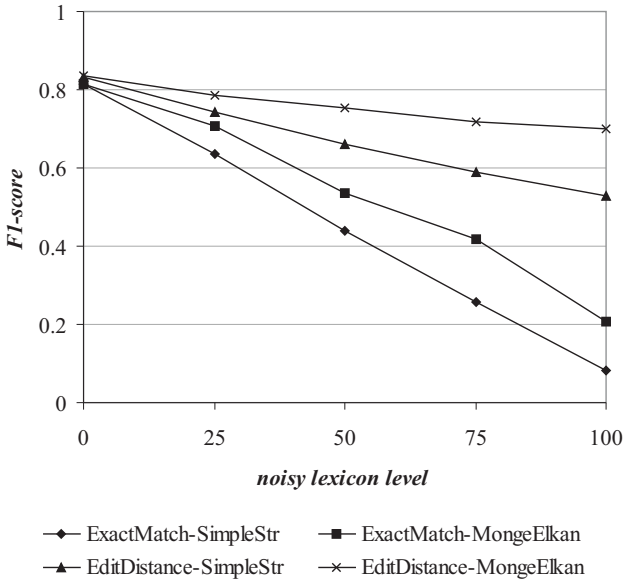


Figure 5.13: Precision/recall/F-measure vs. threshold plots

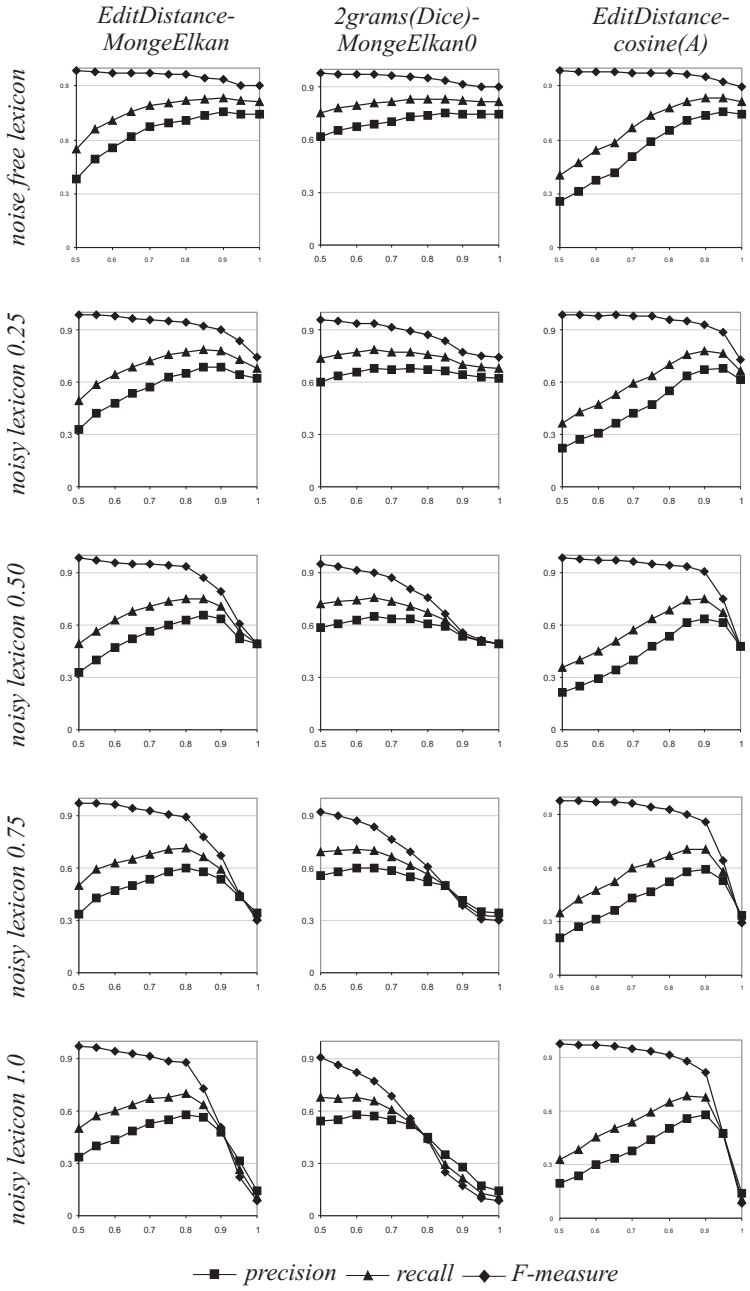


Table 5.7: String matching thresholds for the F1-score results reported in table 5.6.

	ExactMatch	EditDist.	EditDist.	2grams(Dice)	EditDist.
P	MongeElkan	SimpleStr	MongeElkan	MongeElkan	cosine(A)
0	0.85	0.90	0.90	0.85	**0.90**
25	0.65	0.85	0.85	0.65	**0.90**
50	0.65	0.80	0.80	0.65	**0.90**
75	0.50	0.75	0.80	0.60	**0.90**
100	0.50	0.75	0.80	0.50	**0.85**

better fixation of the threshold was *cosine(A)*. This method was proposed in this thesis in section 3.5.

5.5.3 Results for Semantic Relatedness Measures

The implemented semantic relatedness measures mentioned in section 5.3 were evaluated using the following experimental parameters:

- EditDistance-MongeElkan as string matching technique.
- Noise-free laptop lexicon.
- Shortest path as disambiguation strategy.

The F-measure for values of the string similarity threshold in (0,1] are plotted in figure ?. The proposed baseline metric is the simple path length measure, but no measure manages to consistently outperform this baseline. In fact, in the zone where the F-measure reach the F1-score (the maximum) path length surpass all the other measures. On the other hand, the Stetina et al. measure do no seem to be suitable for our information extraction system.

5.5.4 Results for Disambiguation Strategies

Before to present the results for different disambiguation strategies, lets consider first the level of ambiguity of the task. The graph in figure ? shows the number of possible uses/senses for different string matching thresholds using EditDistance-MongeElkan as string comparison method. It is important to remember that the total number of targets in the laptop evaluation corpus is 1596. Thus, with a typical threshold fixed in 0.8 the term disambiguator has to choose 1596 senses among more than 90,000 candidates.

The experimental parameters used to evaluate the implemented disambiguation strategies (section 5.4) are:

- EditDistance-MongeElkan as string matching technique.
- Noise-free laptop lexicon.

Figure 5.14: F-measure using shortest-path and different semantic relatedness measures.

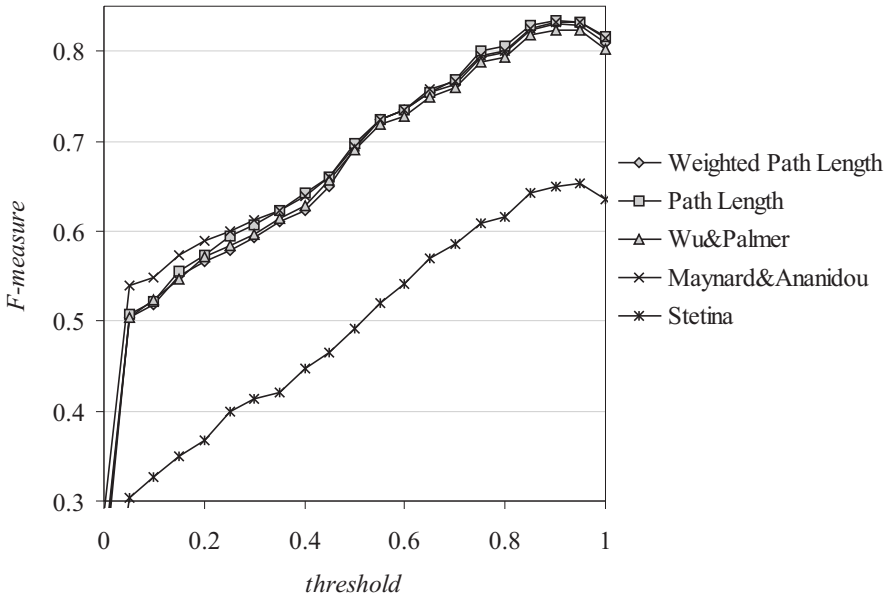


Figure 5.15: Number of nodes in the disambiguation graph for different string comparison thresholds.

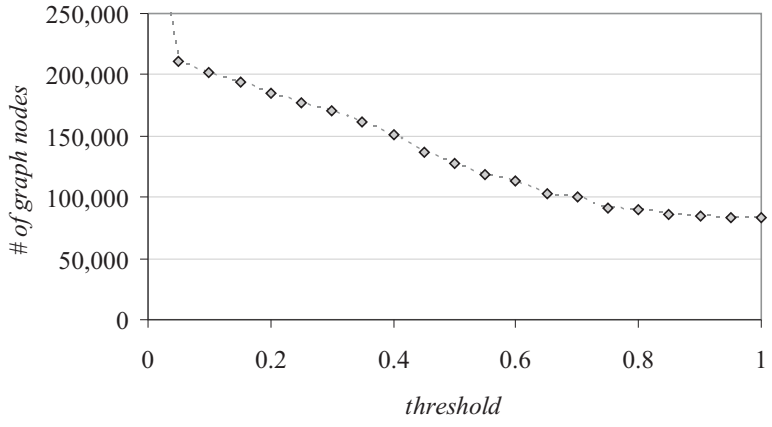
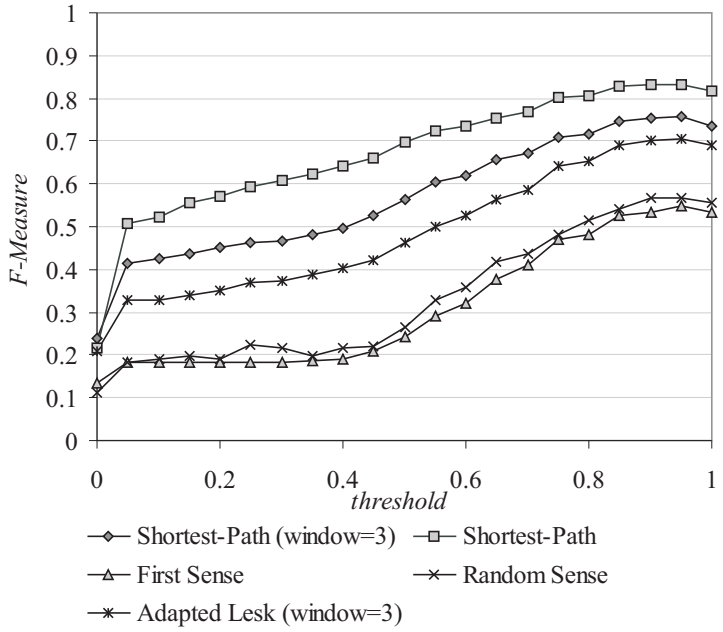


Figure 5.16: Extraction performance (F-measure) comparison between the Adapted Lesk with window size of 3 vs. shortest path and baselines.



- Path length as semantic relatedness measure.

The obtained results can be viewed in the graphic in figure 5.16.

5.6 Conclusions

The results of the experiments related with the string matching methods let us to conclude that the use of the evaluated matching measures surpassed the overall performance of the system in comparison with the use of simplistic exact match. Additionally, it is possible to conclude that when the string matching threshold is greater than 0.8 the additional noise that is added to the disambiguation process do not reduce significantly the overall performance.

Particularly, the methods that combine a character level measure with a token level metric showed the better resilience against noise. Comparable results were obtained using: *EditDistance-MongeElkan*, *2grams(Dice)-MongeElkan* and *EditDistance-cosine(A)*. This last method showed the best stability in the threshold where it reaches the best performance (0.9) in the information extraction task being exposed to different noise levels.

Related to the semantic relatedness experiments, clearly we can conclude that the simple *path length* measure is the more convenient option. Other mea-

sures with more complex approaches reached the same performance that *path length* reached.

Among the disambiguation strategies evaluated, the proposed *shortest path* strategy clearly out performed other techniques and baselines.

Chapter 6

Matching Sensitive to Acronyms and Abbreviations

The chapter 3 studied the problem of compare text strings in a fuzzy way, dealing mainly with problem such as misspellings, typos, token order, morphological variations. The chapter 5 showed that it is possible to provide robustness to an information extraction system using fuzzy string searching. However, there is another important source of variations in strings, which is the fact that in written and spoken language it is very common to compress long names in shorter ones. This practice in the human language has not more than 200 years but nowadays is omnipresent in the modern language. Particularly, the use of acronyms and abbreviations in the data-sheets in the IT domain is very frequent.

However, the reviewed techniques for approximate string matching are not specially designed to deal with the problem of match short-forms and long-forms with the exception of the edit-distance proposed by Gotoh [40]. The Gotoh distance proposes the open and extend gap edition operations with lower costs. Consequently, the editions to transform "USB" into "Universal Serial Bus" are three *open-gap* operations after the letters 'U', 'S', 'B' and 17 *extend-gap* operations. However, this approach can make "false-matches", e.g. consider and "acronym" made with the last letters with the same previous example, that is "lls". The Gotoh distance from "USB" or "lls" to "Universal Serial Bus" is the same.

On the other hand, the results of the previous chapter showed that is more convenient to treat text strings as sequences of tokens instead as sequences of characters, but the token oriented approaches do not seem to be suitable to acronym matching. For instance, matching "USB" with "Universal Serial Bus" at token level only can be possible if the token "USB" is divided in three tokens. The heuristic of separate into tokens consecutive capital letters is a common heuristic used in the *acronym matching* field.

In this chapter we present some existing approaches to the problem of acronym and abbreviation matching. Additionally, a general integrated method

for applying some of acronym heuristics with the matching methods presented in chapter 3. The aim of this integration method is to obtain a string matcher that conserves the properties of the string matching measures and also to support acronym matching. The proposed method is not going to be validated experimentally in a direct way but like part of complete extraction system proposed in this thesis. Thus, the overall extraction performance will be compared with and without the proposed integration method.

This chapter is organized as follows. The section 6.1 present a small survey of the known approaches to address the acronym matching task. A new method to integrate acronym matching heuristics to the token-based string comparison methods is proposed in section 6.2. Finally, the effect of the proposed method in the extraction system presented in chapter ? is discussed in section 6.3.

6.1 Acronym/Abbreviation Matching

The most common form of abbreviation are the acronyms, which that are short-forms made with the initial letters or syllables of the words in the long-form (e.g. "USB" stand for "Universal Serial Bus"). Acronyms are single-word contracted versions of multi-word terms that are used in place of their long versions. Nevertheless, new acronym construction styles are being appearing and the approaches are so "creative" that the task can be compared with that of assigning names to things. There are notices of the earlier use of acronym since the 19th century. Acronyms are being used commonly in modern (post 1940s) English in technical and legal documents and its use has been spread to many other domains and languages nowadays. Commonly, acronyms are not included in general domain dictionaries and " everybody creates new acronyms every day".

Acronyms/matching is the task of compare short-, long- form pairs (i.e. abbreviations-definition) and to identify valid pairs. Acronym detection (a.k.a. acronym identification of acronym recognition) is a task broader than acronym matching, that is finding pairs of acronym and its definition in document corpora in order to build an acronym definition list. This task is usually divided into two sub-tasks: the acronym-definition candidate pairs identification and the *acronym-definition* matching evaluation. We are specially interested in the second one, that is: given an *acronym-definition* pair, to assess how the *acronym* could be generated from the *definition*.

Some applications of the acronym detection task are: OCR refinement, information retrieval (retrieve documents that uses the definition of the acronym when the query only has the acronym), knowledge extraction, thesaurus construction, ontology construction, etc.

The metrics used for acronym detection evaluation are *precision* and *recall* and *F-measure* as is common in many text retrieval experiments. Unfortunately the results reported by authors shows the overall performance of their methods combining candidate detection and matching evaluation. Some recall and precision results are reported for the sake of comparison, but they do not necessarily reflects performance only in the acronym matching task.

$$recall = \frac{\#correct\,acronym\,definitions\,found}{total\,\#\,of\,acronym\,definitions\,in\,document}$$

$$precision = \frac{\#correct\,acronym\,definitions\,found}{total\,\#\,of\,acronym\,definitions\,found}$$

$$F - measure = \frac{2 * precision * recall}{precision + recall}$$

The most common approaches to the acronym/abbreviation detection problem have addressed the task using static pattern matching rules and specific heuristics. We consider as unsupervised those methods because do not uses any kind of labeled data or predefined lists of *acronym-definition* pairs to build their models. Previous work related to the *acronym-definition* matching evaluation sub-task (omitting the candidate identification task) is presented in the following subsections.

6.1.1 String Comparison-Based Methods

The earliest work that we have notice is the Acronym Finding Program (AFP) by Taghva and Gilbreth [86] who used the *LCS* [69] as underlying matching technique. They moved in a sliding window over the document looking for occurrences of acronym-definition pairs obtaining in its experiments precision of 98% at recall level of 86% using governmental documents. Their approach considered only acronyms from three to ten characters. Two letters acronyms such as "US" ("United States") or "AI "("Artificial Intelligence") had too high error rates due to the high relative error of *LCS* measure in short strings. The model was refined using heuristics such as to optionally ignore stop words (e.g. *and*, *of*), to separate hyphenated words (e.g. *gas-cooled*, *non-high-level*), $|acronym|/|definition|$ (the || returns the number of characters of a string) length ratio and first letter matching.

6.1.2 Rule-/Heuristics-Based Methods

Monge and Elkan [64] also proposed a rule matching algorithm for abbreviation matching based in the following four patterns:

1. The abbreviation is a prefix of its expansion, e.g. "Univ." abbreviates "University", or
2. The abbreviation combines a prefix and a suffix of its expansion, e.g. "Dept." matches "Department", or
3. The abbreviation combines a prefix and a suffix of its expansion, e.g. "UCSD" abbreviates "University of California San Diego", or

4. The abbreviation is a concatenation of prefixes from its expansion, e.g. "Caltech" matches "California Institute of Technology"

The authors do not report results for the matching task because it was embedded in a record linkage general task.

Yeates [94]proposed the Three Letter Acronyms (TLA) algorithm, a greedy sequential matching algorithm between the *acronym* and its candidate *definition*, but adding a set of heuristics in order to discard false alarms. The heuristics proposed by Yeates are the following:

1. Acronyms are shorter than their definitions .
2. Acronyms contain initials of most of the words in their definitions.
3. Acronyms are given in upper case.
4. Shorter acronyms tend to have longer words in their definition.
5. Longer acronyms tend to have more stop words.

The matching algorithm tokenizes the *definition* into words. Further, the *definition* is tested from left-to-right trying in order to find a match against the *acronym* using none or the initial letters of each token. Yeates claimed to obtain a more general method than Taghva and Gilbreth's method in spite of the lower performance of his method. Additionally, Yeates proposed as improvement the use of machine learning techniques to weight the heuristics in order to adapt the method to different data sets.

Another rule based acronym extractor was named Acrophile by Larkey et al. [50]. The Acrophile method searches first for acronym candidates in the document using regular expressions. Next, a greedy acronym expansion algorithm is used in the neighboring of the acronym candidates to find possible expansions and weighting each candidate. Larkey et al. combined several rules and heuristics for finding and expanding acronyms.

Pustejovsky et al.[76] noticed that the standard definition of acronym (short forms constructed with initial letters of definition words) is poorly used in domains such as biomedical literature. For instance *carboxifluorescein diacetate* (CFDA) do not constitute an acronym according to the previous definitions and fits better with abbreviation definition. The proposed method named AcroMed, tries to match all acronym characters as a prefix of infix of the words of the candidate definition. Considering only *definitions* that starts with the same character as the acronym a score is computed according to the formula:

$$Score = \frac{\#\,of\ words\ in\ the\ match}{\#\,of\ characters\ in\ the\ acronym}$$

If *Score* is lower than a preestablished threshold (*i.e.* 1.5 had the best result in experiments) then the candidate definition is accepted. This method combined with a simple regular expression strategy to find candidate pairs outperforms the Acrophile algorithm using the same evaluation corpus. Acromed

reached recall of 63% at precision of 90% and the best result obtained by Acrophile was recall of 26% at the same precision.

Schwartz and Hearts [82] proposed a simple but fast and effective method. The matching algorithm process each part of the candidate *acronym* and *definition* form right-to-left trying to match all characters from *acronym* and some characters from *definition*. Only the first characters of *acronym* and *definition* are restricted to match to reach a valid pair. The algorithm do not uses complex heuristics that usually are developed for specific training sets and consequently difficult generalization to other domains and documents. This method reached a recall of 82% at precision of 96% using abstracts from MEDLINE[1], comparable to the results obtained by Chang et al. (83% recall at 80% precision) using a supervised approach and those of Pustejovsky et al. (72% recall at 98% precision) in the same biomedical texts.

The idea of using compression for acronym detection was proposed by Yeates et al. [95]. The method relies in the idea that in valid *definition-acronym* pairs, the *definition* can be coded as the *acronym* with an acronym-specific code using fewer bits than with a general text compression model such as PPM [10]. The decision rule was given establishing a threshold θ compared with the ratio of the number of bits required for coding the acronym model and the compression model.

$$\frac{bits_{acronym\ model}}{bits_{text\ compression\ model}} \leqq \theta$$

The *acronym model* uses a tokenized version of *definition* and stores the words that have been used to build the acronym and the number of initial characters that were used. Yeates et al. reported recall close to 80% at 85%-90% precision with $\theta = 0.2$ using 150 computer science technical reports of 1.4 millions of words including 1080 manually labeled acronym definitions.

Adar [1] proposed an acronym matching method mixing approximate string matching with a set of rules for scoring the matches. The method firstly found all possible common sub-sequences between the candidate abbreviation-definition pair. Further, the following score rules (related to acronym matching) are applied for each candidate pair:

- For every abbreviation character that is at the start of a definition word add 1 to the score.
- The number of definition words should be less than or equal to the number abbreviation characters.
- For every extra word subtract 1.

Empirically, a threshold of 0 was established for valid matching scores. The experimental evaluation of this method was made with a corpus of biomedical texts with 11,253,125 documents. The number of unique abbreviation, definition pairs was 136,082. Of these, a random sample of 644 were manually tested obtaining a 96% accuracy rate.

[1]MEDLINE. http://www.ncbi.nlm.nih.gov/pubmed

Table 6.1: Rule examples of the Park and Byrd's method. [70]

Abbreviation	Abbv. Pattern	Definition	Def. Pattern	Formation Rule
2-MASS	ncccc	Two-Micron All Sky Survey	wwwww	(1,R) (2,F) (3,F) (4,F) (5,F)
CFCs	cccc	chlorofluorocarbons	w	(1,F) (1,I) (1,I) (1,I) (1,I)
X2B	cnc	Hexadecimal to Binary	phsw	(1,R) (3,R) (4,F)
NEXT	cccc	Near-End CrossTalk	www	(1,F) (2,F) (3,R) (3,I)
SN1987A	ccnc	Supernove 1987A	phnw	(1,F) (2,F) (3,R) (4,F)

6.1.3 Induced Rules Methods

Park and Byrd [70] proposed an abbreviation rule discovery system. The training set is a list of valid *abbreviation-definition* pairs where a set of patters and formation rules are extracted and ordered by occurrence. Some examples of extracted rules taken from [70] are shown in table 6.1.

Abbreviation patterns are formed from abbreviations replacing each alphabetic character by a 'c' and each sequence of numeric character (including '.' and ',') by a 'n'. For instance patterns for "2MASS" and "V.3.5" abbreviations are respectively "ncccc" and "cn". Definition patterns are formed with the following code: 'w' (normal word), 's' (stopword form a pre-defined list), 'p' (prefix from a pre-defined list), 'h' (headword *i.e.* remaining without prefix) and 'n' (number). For instance the definition pattern for "Supernova 1987A" is 'phnw'.

Formation rules are lists of pairs (*word position*, *formation method*). Formation method is coded as follows:

F first character of a word

I interior character of a word

L last character of a word

E exact match (only for matching numbers)

R special replacement match *e.g* 'first' for '1'

Once the patterns and formation rules are extracted from training set, frequent rules are selected. Further, for validation of an unseen *abbreviation-definition* pair rules are selected matching abbreviation and definition patterns. Next, if the formation rule can generate the *abbreviation* from the candidate *definition* then the candidate pair is labeled as valid. Park and Byrd tested their method against three documents: a book about automotive engineering, a technical book from a pharmaceutical company, and NASA press releases for 1999. All results of recall and precision measures obtained was in the range over 93.8%-100%. Nevertheless, authors do not give details about the nature and size of the training set.

6.1.4 Supervised Methods

Chang et al.*[18]* proposed a supervised approach to the *abbreviation-definition* pair matching. They extract a vector of 8 features from a *abbreviation-definition* pair using a character alignment between them obtained with the longest common sub-sequence (LCS) algorithm.The features are:

1. Percent of letters in abbreviation in lower case
2. Percent of letters aligned at the beginning of a word
3. Percent of letters aligned at the end of a word
4. Percent of letters aligned on a syllable boundary
5. Percent of letters aligned immediately after another letter
6. Percent of letters in the abbreviation that are aligned
7. Number of words in the *definition* not aligned to the abbreviation
8. Average number of aligned letters per word

Positive examples for the training set were obtained from a list of valid abbreviations and its definitions using the LCS alignments. Negative examples were generated with incorrect alignments of correct abbreviations and correct alignments of incorrect abbreviations. The machine learning model used was Logistic Regression and the best results reached recall of 83% at 80% precision against the Medstract Gold Standard Evaluation Corpus [75] of labeled biomedical abstracts.

Nadeau and Turney [66] proposed a supervised method method similar to the one of Chang et al. but with more features [18]. They considered all possible acronym-definition pairs at sentence level and used a set of heuristics to reduce the search space. The training and evaluation sets were built with the Medstract Gold Standard. Results from experiments using a support vector machine reached *F-measure*=88.3%, close to the performance obtained by Schwartz and Hearst [82] of *F-measure*=88.4%

6.2 A Fuzzy String Matcher sensitive to Acronyms and Abbreviations

The strategy that is proposed in this section aims to integrate some of the heuristics reviewed in this chapter into a token-based approximate string matcher. An idea could be to "activate" the heuristics only when one of the tokens being compared seem to be an acronym. However, it is also desirable that when two acronyms are being compared those heuristics should be "deactivated" and the acronyms should be compared as regular tokens. Additionally, there are some

token separators such as the period that have to be ignored in some cases (e.g. when "U.S." is compared with "US").

The proposed strategy consist of enumerating all possible token configurations obtained when the heuristics are applied or ignored. Further, each pair of token configurations are compared with a token-based string similarity method and the highest obtained value is returned as the similarity measure for the strings being compared. The acronym tokenizing method is presented in section 6.2.1. The used acronym heuristics and the token combination enumeration policy are explained in section 6.2.2.

6.2.1 The Acronym Tokenizer

Additionally to the tokenizer presented in section 6.2.1, an acronym tokenizer is provided for implementing some simple acronym heuristics. This acronym tokenizer is optionally applied only into tokens obtained with the main tokenizer. The goal is to identify possible acronyms and separate as tokens the possible initials letters of the long-form. The used acronym heuristics are:

- Divide as separate tokens two consecutive capital letter characters, e.g. "US" is tokenized as "U", "S".
- Divide as separate tokens a lower-case character followed by a upper-case character, e.g. "miniDIP" is tokenized as "mini", "D", "I", "P".
- Divide as separate token capital letters separated by a period ('.'), e.g. "U.S.A." is tokenized as "U", ".", "S", ".", "A", ".".

Clearly, if none of those rules can be applied to a specific token, it is considered that this token does not contain an acronym.

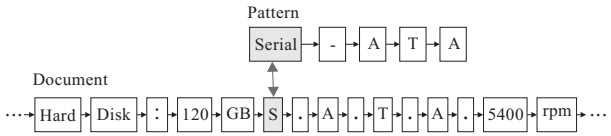
6.2.2 Acronym Sensitive Token Matcher

The proposed acronym sensitive matching strategy consist of enumerate all the possible configurations of token between two pairs of multi-token strings considering and ignoring alternatively some acronym heuristics. The two optional heuristics to be combined are:

- If the actual token can be divided with the acronym tokenizer, then: option 1) consider the tokens obtained with the acronym tokenizer, option 2) consider the original token.
- If the actual token is a period, a hyphen or a slash, then: option 1) ignore it, option 2) consider it.

The enumeration of the combinations is going to be illustrated with an example. Lets consider the pattern and document being compared at the shaded position in figure 6.1. The possible token configurations using the proposed acronym heuristics are enumerated in figure 6.2. Each token configuration is compared

Figure 6.1: Pattern/document example.



with a string matching method in order to obtain a similarity measure. The configuration with the highest similarity is chosen. The example in figure 6.2 has the better configuration in the combination marked with j)*.

6.3 Experimental Results

The method proposed in this chapter was integrated to the information extraction system proposed in chapter 5. The best results obtained in that chapter are compared using the same parameter configuration with the acronym sensitive token matcher integrated. The used configuration used in the information extraction system is:

- *EditDistance-MongeElkan* as string matching method.
- *Path length* as semantic relatedness measure.
- *Shortest path* as term disambiguation strategy.
- The noise-free laptop lexicon was used.

The proposed base line is the F-measure obtained using the same configuration but with exact string comparison.

The results in figure 6.3 show that the use of the acronym sensitive matcher clearly improve the performance of the information extraction system. Additionally, the range of the string matching threshold that is above the baseline was considerably extended.

The approach proposed in this chapter was particularly designed to fit the styles of acronym construction in the data-sheets considered in this thesis. Besides, the acronym sensitive matcher could be a candidate to be a general matching method for acronyms but a comparison with other methods and experiments with other data sets would be mandatory. Although, the obtained results with our particular application are encouraging, the proposed method should be considered as a baseline for a future work.

Figure 6.2: Example of the token configuration combinations enumerated by the acronym sensitive token matcher.

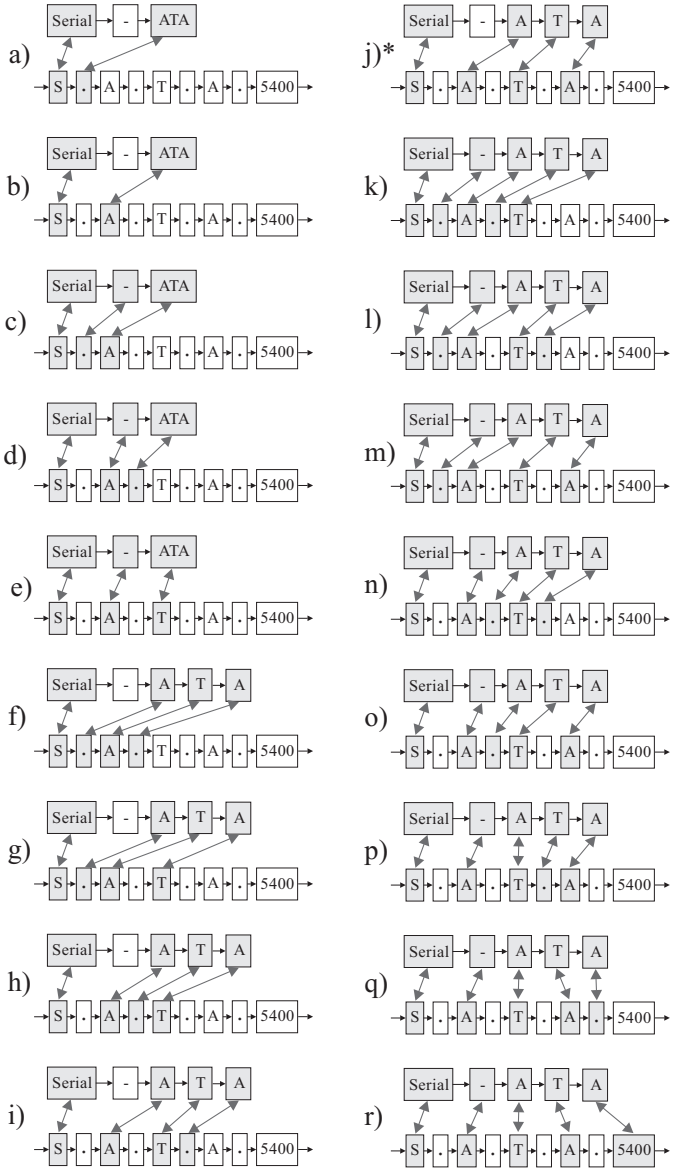
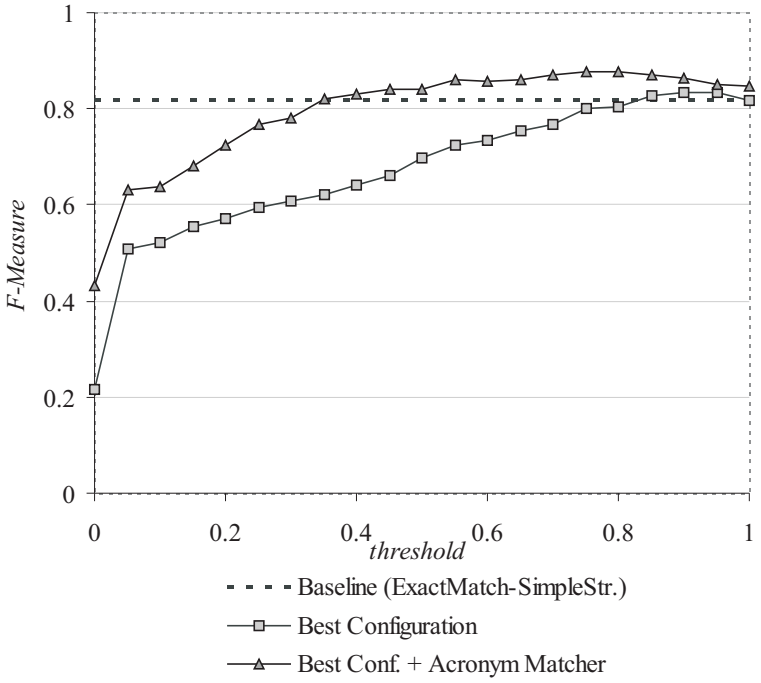


Figure 6.3: Performance comparison of the information extraction system with and without the acronym sensitive matcher.



Chapter 7

Conclusions

7.1 Conclusions

The following enumeration resumes the strongest and most important statements that can be made from our observations:

1. The results obtained in chapter 3let us to conclude that exploiting the fact that text strings are naturally divided into sub-sequences of words or tokens, the quality of the string comparison processes can be increased .

2. The use of approximate string matching techniques that combine strategies at character and at token level, embedded in our specific IE task, bring resilience against noise in spite of the additional ambiguity introduced by their use.

3. Given the results obtained in chapter 5, the Knowledge-Based, WSD-inspired proposed algorithm were capable of handling the IE task in data-rich documents.

4. The proposed acronym sensitive string matcher were capable of increasing the performance of our IE system given the high number of acronyms in the laptop lexicon.

7.2 Summary of Contributions

The contributions of this thesis can be summarized as follows:

1. The proposed methods based on fuzzy cardinality estimation proposed in section 3.5 are very competitive among the static approximate string comparison methods.

2. It was shown that, a IE process can be addressed as a WSD problem when the density of extraction targets and the number of target types are high.

This approach have not been considered in the past, thus our work opens the way to a new set of IE methods based on the WSD ideas.

7.3 Future Work

1. The developed prototype can be the base for the construction of an assisted semantic document annotation system. The data obtained with this tool brings the possibility to build an IE model with predicting abilities. That is, a model that can enrich the ontology and the lexicon.

2. The proposed methods for combining character and token level measures can be extended to integrate other measures in replacement of those of character-level. For instance, if there is evidence that the words "controller" and "adapter" are synonyms in the IT domain, a comparison between "video controller" and "video adapter" can be easily identified as a valid match. Similarly, using general bi-lingual dictionaries the information extraction process can be extended to many languages.

Appendix A

Laptop Ontology

Laptop $\rightarrow_{has-part}$[1] [CommercialOffer, LaptopID, Processor, Chipset, Memory, HardDisk, Display, VideoAdapter, OpticalDrive, NetworkAdapter, Modem, InputDevice, Battery, ACAdapter, AudioAdapter, Interfaces, ExpansionSlots, WarrantyServices, PhysicalDescription, Software, Webcam, MediaAdapter, SecuritySpecs]

A.1 CommercialOffer

CommercialOffer $\rightarrow_{has-part}$ [ListPrice, MonthlyPrice, NetPrice, Rebate, Shipping, Availability, NoPaymentsTime] ListPrice \rightarrow_{has-part} [Money] MonthlyPrice \rightarrow_{has-part} [Money]

NetPrice $\rightarrow_{has-part}$ [Money] Rebate \rightarrow_{has-part} [Money]

Money $\rightarrow_{has-part}$ [Currency, MoneyAmount]

Shipping $\rightarrow_{has-part}$ [Money, TimeMeasurement]

NoPaymentsTime $\rightarrow_{has-part}$ [TimeMeasurement]

A.2 ProductID

LaptopID $\rightarrow_{has-part}$ [UPCCode, Brand, Family, SubFamily, Model, PartNumber, Certification]

ProcessorID $\rightarrow_{has-part}$ [Brand, Family, Model]

VideoAdapterID $\rightarrow_{has-part}$ [Brand, Family, Model]

WirelessAdapterID $\rightarrow_{has-part}$ [Brand, Model]

[1] All "has-part", "has-attribute" and "has-capability" relationships are denoted as "has-part".

BroadBandWirelessAdapterID $\rightarrow_{has-part}$ [Brand, Model]

ModemID $\rightarrow_{has-part}$ [Brand]

AudioAdapterID $\rightarrow_{has-part}$ [Brand, Family]

SpeakerID $\rightarrow_{has-part}$ [Brand]

ChipsetID $\rightarrow_{has-part}$ [Brand, Family, Model]

A.3 Processor

Processor $\rightarrow_{has-part}$ [ProcessorID, BusWidth, FrontSideBus, Cache, Speed, BusWidth, Technology]

FrontSideBus $\rightarrow_{has-part}$ [FrequencyMagnitude, MegaHertz]

Cache $\rightarrow_{has-part}$ [CacheSize, CacheLevel]

CacheSize $\rightarrow_{has-part}$ [MemorySize]

Speed $\rightarrow_{is-a}$ [FrequencyMeasurement]

BusWidth $\rightarrow_{has-part}$ [TwoPower, Bit]

A.4 Chipset & Memory

Chipset $\rightarrow_{has-part}$ [ChipsetID]

Memory $\rightarrow_{has-part}$ [Installed, MaxInstallable, Technology, ModuleTransferRate, ModuleType, MemorySlots, hasECC]

ModuleTransferRate $\rightarrow_{has-part}$ [Name, MemorySpeed]

MemorySpeed $\rightarrow_{has-part}$ [FrequencyMeasurement]

Installed $\rightarrow_{has-part}$ [MemorySize]

MaxInstallable $\rightarrow_{has-part}$ [MemorySize]

MemorySlots $\rightarrow_{has-part}$ [TotalSlots, OccupiedSlots, AvailableSlots]

TotalSlots $\rightarrow_{has-part}$ [Digit]

OccupiedSlots $\rightarrow_{has-part}$ [Digit]

AvailableSlots $\rightarrow_{has-part}$ [Digit]

hasECC $\rightarrow_{has-part}$ [Boolean]

A.5 HardDisk

HardDisk $\rightarrow_{has-part}$ [Capacity, MaxStorageCapacity, Controller, HDRotationSpeed, MotionSensor, MinSeekTime, AverageSeekTime, MaxSeekTime, PhysicalDescription]

Capacity $\rightarrow_{has-part}$ [MemorySize]

MaxStorageCapacity $\rightarrow_{has-part}$ [MemorySize]

HDRotationSpeed $\rightarrow_{has-part}$ [HDRotationSpeedMagnitude, RevolutionsPerMinute]

MinSeekTime $\rightarrow_{has-part}$ [TimeMeasurement]

AverageSeekTime $\rightarrow_{has-part}$ [TimeMeasurement]

MaxSeekTime $\rightarrow_{has-part}$ [TimeMeasurement]

A.6 Display

Display $\rightarrow_{has-part}$ [PixelResolution, NamedResolution, Technology, Diagonal, PixelPitch, ContrastRatio, hasAmbientLightSensor, isWidescreen]

PixelResolution $\rightarrow_{has-part}$ [PixelResolutionHorizontal, X, PixelResolutionVertical]

Diagonal $\rightarrow_{has-part}$ [DistanceMagnitude, Inch]

PixelPitch $\rightarrow_{has-part}$ [DistanceMeasurement]

ContrastRatio $\rightarrow_{has-part}$ [Ratio] hasAmbientLightSensor \rightarrow_{has-part} [Boolean]

isWidescreen $\rightarrow_{has-part}$ [Boolean]

A.7 VideoAdapter

VideoAdapter $\rightarrow_{has-part}$ [VideoAdapterID, VideoMemory, Slot, TVTuner, VideoInterfaces]

VideoInterfaces $\rightarrow_{has-part}$ [HDMI]

VideoMemory $\rightarrow_{has-part}$ [Installed, MaxInstallable, isShared, isDedicated, isDiscrete, Technology]

TVTuner $\rightarrow_{has-part}$ [TVFormat, hasTVTunner]

hasTVTunner $\rightarrow_{has-part}$ [Boolean]

A.8 OpticalDrive

OpticalDrive $\rightarrow_{has-part}$ [OpticalFormat, DVDLayer, Labeling, supportsBluRay, supportsDoubleLayer]

OpticalFormat $\rightarrow_{has-part}$ [CDFormat, DVDFormat, BluRayFormat, HDDVDFormat, numberOfFormats]

numberOfFormats $\rightarrow_{has-part}$ [Integer]

CDFormat $\rightarrow_{has-part}$ [Media, ReadSpeed, WriteSpeed]

DVDFormat $\rightarrow_{has-part}$ [Media, ReadSpeed, WriteSpeed]

BluRayFormat $\rightarrow_{has-part}$ [Media, ReadSpeed, WriteSpeed]

HDDVDFormat $\rightarrow_{has-part}$ [Media]

ReadSpeed $\rightarrow_{is-a}$ [OpticalDriveSpeedMagnitude]

WriteSpeed $\rightarrow_{is-a}$ [OpticalDriveSpeedMagnitude]

supportsBluRay $\rightarrow_{has-part}$ [Boolean] supportsDoubleLayer \rightarrow_{has-part} [Boolean]

A.9 NetworkAdapter (s) & Modem

NetworkAdapter $\rightarrow_{has-part}$ [LANAdapter, WirelessAdapter, BlueToothAdapter, BroadBandWirelessAdapter]

LANAdapter $\rightarrow_{has-part}$ [DataRate, DataRateUnits, Jack, hasLANAdapter]

hasLANAdapter $\rightarrow_{has-part}$ [Boolean]

WirelessAdapter $\rightarrow_{has-part}$ [WirelessAdapterID, Autentication, Version, hasAntenna, hasWirelessAdapter, hasOnOffSwitch]

hasWirelessAdapter $\rightarrow_{has-part}$ [Boolean]

hasOnOffSwitch $\rightarrow_{has-part}$ [Boolean]

BlueToothAdapter $\rightarrow_{has-part}$ [Technology, Version, hasBlueToothAdapter, hasBlueToothAntenna]

hasBlueToothAdapter $\rightarrow_{has-part}$ [Boolean]

hasBlueToothAntenna $\rightarrow_{has-part}$ [Boolean]

BroadBandWirelessAdapter $\rightarrow_{has-part}$ [BroadBandWirelessAdapterID, hasAntenna]

Modem $\rightarrow_{has-part}$ [ModemID, Version, Jack, Certification, isSoftwareModem]

isSoftwareModem $\rightarrow_{has-part}$ [Boolean]

A.10 InputDevice (s)

InputDevice $\rightarrow_{has-part}$ [Keyboard, PointingDevice, HotKeys, hasRemoteControl]

hasRemoteControl $\rightarrow_{has-part}$ [Boolean]

PointingDevice $\rightarrow_{has-part}$ [Technology, hasLeftButton, hasRightButton, hasCenterButton, hasScrollZone, isElectroStaticTouchPad]

hasLeftButton $\rightarrow_{has-part}$ [Boolean]

hasRightButton $\rightarrow_{has-part}$ [Boolean]

hasCenterButton $\rightarrow_{has-part}$ [Boolean]

hasScrollZone $\rightarrow_{has-part}$ [Boolean]

isElectroStaticTouchPad $\rightarrow_{has-part}$ [Boolean]

Keyboard $\rightarrow_{has-part}$ [Country, Layout, KeyPitch, KeyStroke, NumberOfKeys, hasNumericKeyPad, hasScrollBar, isSpillResistant]

hasNumericKeyPad $\rightarrow_{has-part}$ [Boolean]

hasScrollBar $\rightarrow_{has-part}$ [Boolean]

isSpillResistant $\rightarrow_{has-part}$ [Boolean]

NumberOfKeys $\rightarrow_{has-part}$ [Integer]

KeyPitch $\rightarrow_{has-part}$ [DistanceMeasurement]

KeyStroke $\rightarrow_{has-part}$ [DistanceMeasurement]

HotKeys $\rightarrow_{has-part}$ [Type]

A.11 Battery & ACAdapter

Battery $\rightarrow_{has-part}$ [Cells, Technology, Life, Energy, PhysicalDescription, RechargeTime, BatteryWarranty, ElectricalCurrentCapacity, isRechargeable, isRemovable]

Cells $\rightarrow_{has-part}$ [Digit]

Life $\rightarrow_{has-part}$ [TimeMeasurement]

RechargeTime $\rightarrow_{has-part}$ [RechargeTimeOff, RechargeTimeOn]

RechargeTimeOff $\rightarrow_{has-part}$ [TimeMeasurement]

RechargeTimeOn $\rightarrow_{has-part}$ [TimeMeasurement]

Energy $\rightarrow_{has-part}$ [EnergyMeasurement]

BatteryWarranty $\rightarrow_{has-part}$ [TimeMeasurement]

ElectricalCurrentCapacity $\rightarrow_{has-part}$ [CurrentMeasurement]

isRechargeable $\rightarrow_{has-part}$ [Boolean]

isRemovable $\rightarrow_{has-part}$ [Boolean]

ACAdapter $\rightarrow_{has-part}$ [ACInput, DCOutput, PhysicalDescription]

ACInput $\rightarrow_{has-part}$ [MinFrequency, MaxFrequency, MinVoltage, MaxVoltage, PowerRequirements, CurrentRequirements, hasVoltageAutoSensing]

MinFrequency $\rightarrow_{has-part}$ [FrequencyMeasurement]

MaxFrequency $\rightarrow_{has-part}$ [FrequencyMeasurement]

MinVoltage $\rightarrow_{has-part}$ [PotentialMeasurement]

MaxVoltage $\rightarrow_{has-part}$ [PotentialMeasurement]

PowerRequirements $\rightarrow_{has-part}$ [PowerMeasurement, Tolerance]

CurrentRequirements $\rightarrow_{has-part}$ [CurrentMeasurement, Tolerance]

Tolerance $\rightarrow_{has-part}$ [Percentage]

hasVoltageAutoSensing $\rightarrow_{has-part}$ [Boolean]

DCOutput $\rightarrow_{has-part}$ [Voltage, Current, Power]

Voltage $\rightarrow_{has-part}$ [PotentialMeasurement]

Current $\rightarrow_{has-part}$ [CurrentMeasurement]

Power $\rightarrow_{has-part}$ [PowerMeasurement]

A.12 PhysicalDescription

PhysicalDescription $\rightarrow_{has-part}$ [Weight, Color, Material, Dimensions]

Weight $\rightarrow_{has-part}$ [WeightMeasurement]

Dimensions $\rightarrow_{has-part}$ [Height, Width, Depth]

Height $\rightarrow_{has-part}$ [DistanceMeasurement, MinHeight, MaxHeight]

Width $\rightarrow_{has-part}$ [DistanceMeasurement]

Depth $\rightarrow_{has-part}$ [DistanceMeasurement]

MinHeight $\rightarrow_{has-part}$ [DistanceMeasurement]

MaxHeight $\rightarrow_{has-part}$ [DistanceMeasurement]

Webcam $\rightarrow_{has-part}$ [Resolution]

Resolution $\rightarrow_{has-part}$ [Magnitude, PixelUnits]

A.13 AudioAdapter & Speaker

AudioAdapter $\rightarrow_{has-part}$ [AudioAdapterID, Speaker, Sampling, Technology, hasMicrophone, MicrophoneDirection, AudioInterfaces]

hasMicrophone $\rightarrow_{has-part}$ [Boolean]

Sampling $\rightarrow_{has-part}$ [Integer, Bit]

AudioInterfaces $\rightarrow_{has-part}$ [HDMI, hasS/PDIF, HeadphoneJack, MicrophoneJack, LineOutJack]

HeadphoneJack $\rightarrow_{has-part}$ [hasHeadphoneJack, AudioJackType]

MicrophoneJack $\rightarrow_{has-part}$ [hasMicrophoneJack, AudioJackType]

LineOutJack $\rightarrow_{has-part}$ [hasLineOutJack, AudioJackType]

hasS/PDIF $\rightarrow_{has-part}$ [Boolean]

hasHeadphoneJack $\rightarrow_{has-part}$ [Boolean]

hasMicrophoneJack $\rightarrow_{has-part}$ [Boolean]

hasLineOutJack $\rightarrow_{has-part}$ [Boolean]

Speaker $\rightarrow_{has-part}$ [SpeakerID, Type, SoundPower, Impedance, hasSpeaker]

SoundPower $\rightarrow_{has-part}$ [PowerMeasurement]

Impedance $\rightarrow_{has-part}$ [ImpedanceMeasurement]

hasSpeaker $\rightarrow_{has-part}$ [Boolean]

A.14 MediaAdapter & Interfaces & Expansion-Slots

MediaAdapter $\rightarrow_{has-part}$ [Technology, NumberOfMediaFormats, hasMediaAdapter]

hasMediaAdapter $\rightarrow_{has-part}$ [Boolean]

NumberOfMediaFormats $\rightarrow_{has-part}$ [Integer]

Interfaces $\rightarrow_{has-part}$ [USB, IEEE1394, hasSerial, hasParallel, hasInfraRed, HDMI, hasPS/2, hasPortReplicatorConnector]

USB $\rightarrow_{has-part}$ [NumberOfPorts, Version]

IEEE1394 $\rightarrow_{has-part}$ [NumberOfPorts, PinOut]

PinOut $\rightarrow_{has-part}$ [Integer, Pin]

HDMI $\rightarrow_{has-part}$ [NumberOfPorts, Version]

VGA $\rightarrow_{has-part}$ [NumberOfPorts, PinOut]

hasSerial $\rightarrow_{has-part}$ [Boolean]

hasParallel $\rightarrow_{has-part}$ [Boolean]

hasInfraRed $\rightarrow_{has-part}$ [Boolean]

hasPS/2 $\rightarrow_{has-part}$ [Boolean]

hasPortReplicatorConnector $\rightarrow_{has-part}$ [Boolean]

ExpansionSlots $\rightarrow_{has-part}$ [PCCardSlots, PCExpressSlots, PCISlots]

PCCardSlots $\rightarrow_{has-part}$ [SlotsNumber, Type, hasPCCardSlots]

PCExpressSlots $\rightarrow_{has-part}$ [SlotsNumber, Type, hasPCExpressSlots]

PCISlots $\rightarrow_{has-part}$ [SlotsNumber, hasPCISlots]

hasPCCardSlots $\rightarrow_{has-part}$ [Boolean]

hasPCExpressSlots $\rightarrow_{has-part}$ [Boolean]

hasPCISlots $\rightarrow_{has-part}$ [Boolean]

SlotsNumber $\rightarrow_{has-part}$ [Digit]

A.15 WarrantyServices

WarrantyServices $\rightarrow_{has-part}$ [Warranty, Support, hasDamageCoverage]

hasDamageCoverage $\rightarrow_{has-part}$ [Boolean]

Warranty $\rightarrow_{has-part}$ [OnSiteWarranty, CarryInWarranty, PartsWarranty, LaborWarranty]

OnSiteWarranty $\rightarrow_{has-part}$ [TimeMeasurement, Region]

CarryInWarranty $\rightarrow_{has-part}$ [TimeMeasurement, Region]

PartsWarranty $\rightarrow_{has-part}$ [TimeMeasurement, Region]

LaborWarranty $\rightarrow_{has-part}$ [TimeMeasurement, Region]

Support $\rightarrow_{has-part}$ [EmailSupport, TelephoneSupport, OnlineSupport]

EmailSupport $\rightarrow_{has-part}$ [EmailAddress, TimeMeasurement]

TelephoneSupport $\rightarrow_{has-part}$ [ToolFreeNumber, Region, WeeklySchedule, TimeMeasurement]

ToolFreeNumber $\rightarrow_{has-part}$ [Integer]

WeeklySchedule $\rightarrow_{has-part}$ [hoursDay, daysWeek]

hoursDay $\rightarrow_{has-part}$ [Integer, Hour]

daysWeek $\rightarrow_{has-part}$ [Integer, Day]

OnlineSupport $\rightarrow_{has-part}$ [URLAddress, TimeMeasurement]

A.16 Software

Software $\rightarrow_{has-part}$ [OperatingSystem, ProductivitySoftware, PhotoSoftware, BurningSoftware, SecuritySoftware, UtilitySoftware, MediaSoftware, VideoSoftware]

OperatingSystem $\rightarrow_{has-part}$ [ProcessorArchitecture, Brand, Family, SoftwareVersion, SubVersion]

ProcessorArchitecture $\rightarrow_{has-part}$ [ProcessorType, BusWidth]

ProductivitySoftware $\rightarrow_{has-part}$ [Brand, Family, SoftwareVersion, SubVersion, Licence]

PhotoSoftware $\rightarrow_{has-part}$ [Brand, Family, SoftwareVersion, SubVersion, Licence]

VideoSoftware $\rightarrow_{has-part}$ [Brand, Family, SoftwareVersion, SubVersion, Licence]

BurningSoftware $\rightarrow_{has-part}$ [Brand, Family, SoftwareVersion, SubVersion, Licence]

SecuritySoftware $\rightarrow_{has-part}$ [Brand, Family, SoftwareVersion, SubVersion, Licence]

UtilitySoftware $\rightarrow_{has-part}$ [Brand, Family, SoftwareVersion, SubVersion]

MediaSoftware $\rightarrow_{has-part}$ [Brand, Family, SoftwareVersion, SubVersion]

SoftwareVersion $\rightarrow_{has-part}$ [Decimal, Integer, Version]

Licence $\rightarrow_{has-part}$ [LicenceType, TrialPeriod]

TrialPeriod $\rightarrow_{has-part}$ [TimeMeasurement]

A.17 Security

SecuritySpecs $\rightarrow_{has-part}$ [PhysicalSecurity, SoftwareSecurity]

PhysicalSecurity $\rightarrow_{has-part}$ [hasCableLockSlot, hasFingerprintSensor, TPM]

TPM $\rightarrow_{has-part}$ [Brand, Version]

hasFingerprintSensor $\rightarrow_{has-part}$ [Boolean]

hasCableLockSlot $\rightarrow_{has-part}$ [Boolean]

SoftwareSecurity $\rightarrow_{has-part}$ [hasPasswordSecurity, hasUserPassword, hasSupervisorPassword]

hasPasswordSecurity $\rightarrow_{has-part}$ [Boolean]

hasUserPassword $\rightarrow_{has-part}$ [Boolean]

hasSupervisorPassword $\rightarrow_{has-part}$ [Boolean]

A.18 Measurements

FrequencyMeasurement $\rightarrow_{has-part}$ [Magnitude, FrequencyUnits]

MemorySize $\rightarrow_{has-part}$ [MemoryMagnitude, MemoryUnits]

TimeMeasurement $\rightarrow_{has-part}$ [Magnitude, TimeUnits]

DistanceMeasurement $\rightarrow_{has-part}$ [Magnitude, DistanceUnits]

EnergyMeasurement $\rightarrow_{has-part}$ [Magnitude, EnergyUnits]

WeightMeasurement $\rightarrow_{has-part}$ [Magnitude, WeightUnits]

PotentialMeasurement $\rightarrow_{has-part}$ [Magnitude, PotentialUnits]

PowerMeasurement $\rightarrow_{has-part}$ [Magnitude, PowerUnits]

CurrentMeasurement $\rightarrow_{has-part}$ [Magnitude, CurrentUnits]

ImpedanceMeasurement $\rightarrow_{has-part}$ [Magnitude, ImpedanceUnits]

CurrentMeasurement $\rightarrow_{has-part}$ [Magnitude, CurrentUnits]

DataRateMeasurement $\rightarrow_{has-part}$ [Magnitude, DataRateUnits]

MemoryMagnitude $\rightarrow_{is-a}$ [TwoPower, Integer, Decimal]

Magnitude $\rightarrow_{is-a}$ [Integer, Decimal, Fraction, Digit]

MoneyAmount $\rightarrow_{is-a}$ [Integer, Decimal]

Percentage $\rightarrow_{has-part}$ [Magnitude, PercentageCharacter]

FrequencyUnits $\rightarrow_{is-a}$ [Hertz, KiloHertz, MegaHertz, GigaHertz]

MemoryUnits $\rightarrow_{is-a}$ [Bit, Byte, KiloByte, MegaByte, GigaByte, TeraByte]

TimeUnits $\rightarrow_{is-a}$ [PicoSecond, NanoSecond, MicroSecond, MiliSecond, Second, Minute, Hour, Day, BusinessDay, Week, Month, Year]

DistanceUnits $\rightarrow_{is-a}$ [PicoMeter, NanoMeter, MicroMeter, MiliMeter, CentiMeter, Inch, Feet, Yard, Meter, KiloMeter, Mile]

EnergyUnits $\rightarrow_{is-a}$ [WattHour, KiloWattHour, MiliWattHour]

WeightUnits $\rightarrow_{is-a}$ [Gram, MiliGram, KiloGram, Pound, Ounce]

PotentialUnits $\rightarrow_{is-a}$ [Volt, MiliVolt, KiloVolt]

PowerUnits $\rightarrow_{is-a}$ [Watt, MiliWatt, KiloWatt, HorsePower]

CurrentUnits $\rightarrow_{is-a}$ [Ampere, MiliAmpere]

PixelUnits $\rightarrow_{is-a}$ [Pixel, MegaPixel]

ImpedanceUnits $\rightarrow_{is-a}$ [Ohm, MiliOhm, KiloOhm, MegaOhm]

DataRateUnits $\rightarrow_{is-a}$ [KilobitPerSecond, MegabitPerSecond, GigabitPerSecond, TerabitPerSecond, KilobytePerSecond, MegabytePerSecond, GigabytePerSecond, TerabytePerSecond]

Fraction $\rightarrow_{has-part}$ [Integer, Slash, Integer]

Ratio $\rightarrow_{has-part}$ [Integer, Colon, Integer]

Boolean $\rightarrow_{is-a}$ [Yes, Not]

Appendix B

Laptop Lexicon

Ampere: [“A”, “Ah”, “Amperes"]

AudioAdapterID_Brand: [“Creative"]

AudioAdapterID_Family: [“Sound Blaster"]

AudioAdapter_Technology: [“SRS Labs audio enhancements", “SRS WOWâ„¢ HD stereo"]

AudioJackType: [“mono", “monaural", “MonoAural", “stereo", “dolby"]

Availability: [“In Stock"]

AvailableSlots: [“Available", “Free", “Disponible"]

Battery_Technology: [“Ion", “Li-ion", “Lithium Ion", “NiCad", “Nickel Cadmium", “NiCd", “NiMH", “Nickel Metal Hydride", “Lithium Polymer"]

Bit: [“Bit", “Bits"]

BluRayFormat_Media: [“BD-R", “BD-R DL", “BD-RE", “BD-RE DL", “BR-ROM"]

BlueToothAdapter_Technology: [“Enhanced Data Rate", “EDR+"]

BlueToothAdapter_Version: [“2.0"]

BroadBandWirelessAdapterID_Brand: [“Verizon", “AT&T", “Sprint"]

BroadBandWirelessAdapterID_Model: [“V740"]

BurningSoftware_Brand: [“Nero", “Sonic", “Roxio", “Sony", “Toshiba"]

BurningSoftware_Family: [“Burning ROM", “Burn", “Creator", “CD Burning", “DVD Burning", “Disc Creator", “Click to DVD", “DVD Creation DVgate"]

BusinessDay: ["business days", "weekdays", "work day", "workday", "working day", "monday to friday", "mon to fri", "mon-fri"]

Byte: ["Byte", "Bytes", "Octet"]

CDFormat_Media: ["CD+(E)G", "CD-DA", "CD-EXTRA", "CD-I", "CD-I Bridge", "CD-MIDI", "CD-R", "CD-ROM", "CD-ROM XA", "CD-RW", "CD-RW", "CD-TEXT", "Multisession CD", "Photo-CD", "Portfolio", "Video CD"]

CacheLevel: ["Level2", "L2"]

CentiMeter: ["centimeter", "cm"]

ChipsetID_Brand: ["ATI", "Intel"]

ChipsetID_Family: ["RADEON"]

ChipsetID_Model: ["XPress 200M", "200M", "945GMS"]

Colon: [":"]

Color: ["Black", "Gray", "White", "Blue", "Carbon", "Silver", "Aluminium Silver", "Cloud", "Red", "Titanium Silver", "Magnesium alloy", "Charcoal Black", "Sienna", "Slate Blue", "Onyx Blue Metallic", "Mist Gray", "Smart Indigo"]

Currency: ["$", "US$", "U$", "Â£", "Â¥", "â,¤", "â,§", "â,£", "â,¬"]

DVDFormat_Media: ["DVDÂ±R/RW", "DVD+R", "DVD+R DL", "DVD+RW", "DVD+-RW", "DVD-10", "DVD-18", "DVD-5", "DVD-9", "DVD-R", "DVD-R DL", "DVD-RAM", "DVD-RAM", "DVD-ROM", "DVD-RW", "HD DVD-ROM", "DVD-Video", "DVD-Audio"]

DVDLayer: ["Single", "Double", "+/-R double", "Double Layer"]

Day: ["day", "days"]

Diagonal_DistanceMagnitude: ["11.1", "12.1", "13.3", "15.4", "17", "4.5", "9.5", "17.0"]

Digit: ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine"]

Display_Technology: ["XBRITE", "BrightView", "LCD", "TFT", "TFT active-matrix", "TruBrite", "Ultra BrightView", "LED backlight", "Blacklit", "active matrix"]

Feet: ["feet", "ft"]

Fraction: ["â...", "â...", "â...›", "â...œ", "5⁄8", "â...ž", "Â¼", "Â½", "Â¾"]

FrontSideBus_FrequencyMagnitude: [“533", “667", “800"]

GigaByte: [“GB", “GByte", “GigaByte", “Giga Byte", “G.B."]

GigaHertz: [“GHz", “Gigahertz", “Giga Hertz"]

GigabitPerSecond: [“Gigabit per second", “Gbit/s", “Gbps"]

GigabytePerSecond: [“Gigabyte per second", “GB/s", “GBps"]

Gram: [“g", “grms", “grams"]

HDDVDFormat_Media: [“HD DVD-ROM", “HD DVD-Video"]

HDMI_Version: [“1.0", “1.1", “1.2", “1.2a", “1.3", “1.3a", “1.3b"]

HDRotationSpeed_HDRotationSpeedMagnitude: [“4200", “5400", “7200"]

HardDisk_Controller: [“ATA", “SATA", “S-ATA", “Serial-ATA", “Serial ATA", “UATA", “U-ATA", “Ultra-ATA", “Ultra ATA", “IDE", “EIDE", “E-IDE", “Enhanced IDE", “Enhanced IDE"]

Hertz: [“Hz", “Hertz"]

HorsePower: [“hp", “horsepower", “horse power"]

HotKeys_Type: [“Media", “Application", “Windows", “Internet", “CD/DVD", “DVD/CD", “DVD", “Play Menu", “Sound", “Volume", “volume up", “volume down", “volume mute", “mute", “Screen Blank", “Security", “Pause", “Play", “Skip", “Skip to Next Track", “Next Track", “Skip to Previous Track", “Previous Track", “Stop", “Instant On"]

Hour: [“hour", “hours", “hrs", “hr"]

Inch: [“inch", “inches", “in", “"]

Keyboard_Country: [“ES", “LA", “FR", “US"]

Keyboard_Layout: [“QWERTY", “DVORAK", “QUERTZ", “AZERTY"]

KiloByte: [“KB", “KByte", “KiloByte", “Kilo Byte", “K.B.", “K"]

KiloGram: [“Kg", “kg", “kilograms"]

KiloHertz: [“KHz", “Kilohertz", “Kilo Hertz"]

KiloMeter: [“kilometer", “km"]

KiloOhm: [“kilohms", “kOhm", “KÎ©"]

KiloVolt: [“KV", “kilovolts"]

KiloWatt: [“KW", “kilowatts"]

KiloWattHour: ["KWh", "KWHr"]

KilobitPerSecond: ["Kilobit per second", "Kbit/s", "Kbps"]

KilobytePerSecond: ["Kilobyte per second", "KB/s", "KBps"]

LANAdapter_DataRate: ["10", "100", "1000", "10/100", "100/1000", "10/100/1000", "Base-T", "Base-TX", "Gigabit Ethernet", "Ethernet", "Fast Ethernet"]

LANAdapter_Jack: ["RJ-45"]

LaptopID_Brand: ["Toshiba", "HP", "Hewlett Packard", "Compaq", "HP Compaq", "Dell", "Sony", "Lenovo", "NEC", "Packard Bell", "Samsung", "LG", "ASUS", "Gateway"]

LaptopID_Certification: ["Australia / NZ A-Tick", "Blue Angel", "BNCI or BELUS", "BSMI", "CCC", "CE Marking", "CI", "CSA", "Energy Star", "FCC", "GOST", "GS", "ICES", "Japan VCCI", "MIC", "SABS", "Saudi Arabian (ICCP)", "UKRSERTCOMPUTER", "UL "]

LaptopID_Family: ["VAIO", "Pavilion", "Presario", "Tecra", "Satellite", "Portege"]

LaptopID_Model: ["A130-ST1311", "dv9500t", "VGN-UX390N", "VGN-TXN15P/B", "TXN15P/B", "U200", "U205", "A135-S4498", "A135", "dv9330us"]

LaptopID_PartNumber: ["PSAD6U-01500J", "PSAD0U-03V011", "RV112UA#ABA"]

LaptopID_SubFamily: ["UX"]

LicenceType: ["Full", "Trial", "Freeware", "GLP", "Shareware", "Try", "Tryware"]

Material: ["Magnersium alloy", "Polycarbonate ABS", "Carbon Fiber"]

MediaAdapter_Technology: ["Memory Stick", "Memory Stick PROâ„¢", "Memory Stickâ„¢", "miniSDâ„¢", "Multimedia Media Card", "SD", "SDIO", "Secure Digital", "Smart Card", "xD Picture Card"]

MediaSoftware_Brand: ["Microsoft", "Real", "Apple", "InterVideo", "Toshiba", "HP"]

MediaSoftware_Family: ["WinDVD", "RealPlayer", "QuickTime", "Windows Media Player", "Media Player", "Speech System", "Rhapsody", "Real-Rhapsody", "Real Rhapsody", "Quick Play"]

MegaByte: ["MB", "MByte", "MegaByte", "Mega Byte", "M.B.", "Megas", "M"]

MegaHertz: ["MHz", "Megahertz", "Mega Hertz"]

MegaOhm: ["megaohms", "MOhm", "MÎ©"]

MegaPixel: ["M", "MP", "megapixel"]

MegabitPerSecond: ["Megabit per second", "Mbit/s", "Mbps"]

MegabytePerSecond: ["Megabyte per second", "MB/s", "MBps"]

Memory_Technology: ["DDR", "DDR2", "DDR3", "EDO", "FPM", "GDDR2", "GDDR3", "GDDR4", "GDDR5", "Rambus", "SDR", "SDRAM", "SGRAM", "XDR", "DDR2-400", "DDR2-533", "DDR2-667", "DDR2-800", "DDR2-1066"]

Meter: ["meter", "m"]

MicroMeter: ["micrometer", "Î¼m"]

MicroSecond: ["Micro Second", "Î¼s"]

MicrophoneDirection: ["directional", "omni-directional"]

Mile: ["mile", "mi"]

MiliAmpere: ["mA", "mAh", "miliamperes"]

MiliGram: ["mg", "miligrams"]

MiliMeter: ["milimeter", "mm"]

MiliOhm: ["miliohms", "mOhm", "mÎ©"]

MiliSecond: ["Milli Second", "ms"]

MiliVolt: ["mV", "milivolts"]

MiliWatt: ["mW", "miliwatts"]

MiliWattHour: ["mWh", "mWHr"]

Minute: ["minutes", "min", "'", "m"]

ModemID_Brand: ["Conexant", "Toshiba"]

Modem_Certification: ["Australian ACA", "C.I.S.P.R.22", "Canadian ICES-003 Class B", "CCIB", "CE Mark", "CSA", "CTR21", "FCC Part 15 Class B", "FCC Part 68", "Industry Canada", "NEMKO", "UL"]

Modem_Jack: ["RJ-11", "Standard Telephone Jack", "Phone Jack"]

Modem_Version: ["56K", "v.90", "v.92"]

ModuleTransferRate_Name: ["PC66", "PC100", "PC133", "PC166", "PC200", "PC1600", "PC2100", "PC2700", "PC3200", "PC4200", "PC5300", "PC2-3200", "PC2-4200", "PC2-5300", "PC-66", "PC-100", "PC-133", "PC-166", "PC-200", "PC-1600", "PC-2100", "PC-2700", "PC-3200", "PC-4200", "PC-5300"]

ModuleType: [“Dimm", “Simm"]

Money: [“Free", “No Charge"]

Month: [“month", “mth"]

MotionSensor: [“3D Accelerometer", “3D DriveGuard", “G-Sensor"]

NamedResolution: [“SXGA+", “Wide SVGA", “Widescreen XGA", “Widescreen XGA+", “WSXGA+", “WUXGA", “WXGA", “WXGA+"]

NanoMeter: [“nanometer", “nm"]

NanoSecond: [“Nano Second", “ns"]

Not: [“No", “Not", “Non", “Optional", “Select models", “Selected models", “some models", “in selected models", “in selected configurations"]

NumberOfMediaFormats: [“formats", “in 1"]

NumberOfPorts: [“one", “two", “three", “four", “five", “six", “1", “2", “3", “4", “5", “6", “1x", “2x", “3x", “4x", “5x", “6x", “x1", “x2", “x3", “x4", “x5", “x6"]

OccupiedSlots: [“Used", “Occupied"]

Ohm: [“Ohms", “Ohm", “Î©"]

OperatingSystem: [“Windows XP", “Win XP", “Windows Vista", “Linux"]

OperatingSystem_Brand: [“Microsoft", “MS", “SUN", “Apple", “Caldera", “SCO", “IBM", “Novell", “Caldera", “Debian", “Mandriva", “Red Hat", “Fedora", “SuSE", “Ubuntu"]

OperatingSystem_Family: [“Windows", “GNU/Linux", “Linux", “Solaris", “MacOS", “Macintosh", “Unix", “SCO Unix", “AIX", “Netware", “FreeDOS"]

OperatingSystem_SoftwareVersion: [“XP", “Vista"]

OpticalDriveSpeedMagnitude: [“2", “2.4", “4", “8", “10", “24", “32", “40", “52"]

OpticalDrive_Labeling: [“DiscT@2", “Labelflashâ„¢", “LightScribe"]

Ounce: [“oz", “ounces"]

PCCardSlots_Type: [“I", “II", “III", “IV"]

PCExpressSlots_Type: [“ExpressCard/34", “34", “ExpressCard/54", “54"]

PercentageCharacter: [“%"]

PhotoSoftware_Brand: [“Corel", “HP"]

PhotoSoftware_Family: [“Photo Album", “Paint Shop", “PhotoSmart"]

PicoMeter: [“picometer", “pm"]

PicoSecond: [“Pico Second", “ps"]

Pin: [“pin", “Pin"]

Pixel: [“pixel", “pixels"]

PixelResolutionHorizontal: [“1024", “1280", “1366", “1400", “1440", “1680", “1680", “1920", “2048"]

PixelResolutionVertical: [“600", “768", “800", “1050", “900", “1050", “1200", “1536"]

PointingDevice_Technology: [“Accupoint", “Dual Mode Pad", “Duo Mode Padâ„¢", “Pointer", “PointStick", “TouchPad", “TrackBall"]

Pound: [“pound", “lb", “lbs", “#"]

ProcessorArchitecture_ProcessorType: [“386", “x86", “i386", “ALPHA", “MIPS", “RISC"]

ProcessorID_Brand: [“Intel", “AMD", “Motorola"]

ProcessorID_Family: [“Pentium", “Celeron", “Celeron M", “Centrino", “Centrino Duo", “Centrino Pro", “Core(TM) 2 Duo", “Core Duo", “Core Solo", “Core 2 Duo", “Pentium dual-core", “Semprom", “Turion", “Turion 64", “Turion 64 Mobile", “Turion 64 X2"]

ProcessorID_Model: [“430", “3200+", “3400+", “3500+", “3600+", “360M", “3800+", “T2060", “T2080", “T2250", “T2350", “T2400", “T2450", “T2500", “T2600", “T5200", “T5300", “T5500", “T5600", “T5600", “T7100", “T7200", “T7300", “T7400", “T7500", “T7600", “T7700", “U1400", “U1500"]

Processor_Technology: [“ULV", “Ultra Low Voltage", “Mobile"]

ProductivitySoftware_Brand: [“Microsoft", “MS", “Corel", “SUN", “Lotus", “IBM", “Adobe"]

ProductivitySoftware_Family: [“Wordperfect", “Office", “Word", “Excel", “Works", “Office X3", “Acrobat Reader", “PowerPoint", “Accounting Express", “Outlook", “Publisher", “Access", “Money"]

REF_ACAdapter: [“AC Adapter", “AC-Adapter", “Power", “external AC Adapter", “Power Supply", “Power"]

REF_ACInput: [“Input"]

REF_AudioAdapter: ["Audio Adapter", "Audio", "Audio Card", "Sound", "Sound Card", "Audio System", "Sound System", "Windows Sound System"]

REF_AudioJackType: ["Jack", "Port", "Jack Type", "Audio Jack", "Plug", "Audio Plug"]

REF_Availability: ["Availability"]

REF_AverageSeekTime: ["Seek Time", "Average Seek Time"]

REF_Battery: ["Battery", "Battery pack", "Primary battery", "Rechargable battery", "Power", "Main Battery"]

REF_BatteryWarranty: ["Warranty"]

REF_Battery_Cells: ["cell", "cells"]

REF_Battery_Life: ["Battery Life", "Estimated Battery Life", "up to"]

REF_Battery_Technology: ["Material", "Type"]

REF_BluRayFormat: ["Blu-ray", "Bluray"]

REF_BlueToothAdapter: ["Bluetooth"]

REF_Brand: ["Brand", "Builder", "Manufacturer", "Made by", "Â®"]

REF_BroadBandWirelessAdapter: ["WWAN", "Wireless WAN", "Broadband Wireless", "Mobile", "Mobile Internet", "Broadband"]

REF_BurningSoftware: ["Burning", "CD authoring", "DVD authoring"]

REF_CDFormat: ["CD", "CompactDisk", "Compact Disk"]

REF_Cache: ["Cache", "Microprocessor Cache", "Processor Cache", "System Cache", "Cache Size"]

REF_CacheLevel: ["Cache Level", "Level"]

REF_CarryInWarranty: ["Carry-in", "Carry in", "at service center"]

REF_Chipset: ["Chipset"]

REF_Color: ["Color", "Case Color", "Chasis Color", "Cover Color"]

REF_Computer: ["Computer", "PC", "Personal Computer"]

REF_ContrastRatio: ["Contrast Ratio"]

REF_DCOutput: ["Output"]

REF_DVDFormat: ["DVD", "Digital Versatile Disc", "Digital Video Disc"]

REF_DVDLayer: ["Layer"]

REF_Depth: ["Depth", "D", "L", "Length"]

REF_Dimensions: ["Dimensions", "Measurement"]

REF_Display: ["Display", "Monitor", "Screen", "Panel"]

REF_Display_Diagonal: ["Diagonal", "Size", "Viewing angle"]

REF_ElectricalCurrentCapacity: ["Capacity"]

REF_EmailSupport: ["Email Support", "Email-support"]

REF_Energy: ["Energy"]

REF_ExpansionSlots: ["Expansion Slot", "Slot", "Slots"]

REF_Family: ["Product family", "Family", "Product Series", "Series", "Product Type", "Type", "Â®"]

REF_FrontSideBus: ["FSB", "Front Side Bus", "System Bus", "system bus running", "up to"]

REF_HDDVDFormat: ["HD DVD", "HDDVD", "High Definition DVD"]

REF_HDMI: ["HDMI", "High Definition Multimedia Interface"]

REF_HDRotationSpeed: ["Speed"]

REF_HardDisk: ["Hard Disk", "HD", "HDD", "Hard Disk Drive", "Disk", "Hard Drive"]

REF_HardDisk_Capacity: ["Capacity", "Size"]

REF_HardDisk_Controller: ["Controller", "Drive Controller", "Interface"]

REF_Height: ["H", "Height", "thick", "thin"]

REF_HotKeys: ["Hot Keys", "Hot Key Functions", "One-Touch Button", "One-Touch Productivity Button", "Launch Button", "Action Button", "Key Function", "Programmable Key", "Dial Control", "Buttons", "Control Button"]

REF_IEEE1394: ["Firewire", "i.Link", "IEEE 1394a", "iLink", "IEEE 1394"]

REF_ITProduct: ["Product", "Product Description", "Product Specs", "Product Specifications"]

REF_InputDevice: ["Input Device"]

REF_Interfaces: ["Ports", "interface", "Interfaces", "Input and Outputs", "Data Interface", "Data ports", "Ports and Connectors", "External Ports"]

REF_KeyStroke: ["Stroke", "Key Stroke"]

REF_Keyboard: ["Keyboard"]

REF_Keyboard_Country: ["Country", "Layout"]

REF_Keyboard_KeyPitch: ["Pitch", "Key Pitch", "Key Size", "center-to-center spacing"]

REF_LANAdapter: ["LAN", "Local Area Network", "Network", "Network Adapter", "Network Connection", "NIC", "Network Interface Card", "Ethernet"]

REF_LANAdapter_DataRate: ["Speed", "Data Rate"]

REF_LANAdapter_Jack: ["Jack", "Port", "Connector", "LAN Port", "Network Port", "Ethernet Port"]

REF_Labeling: ["Labeling"]

REF_LaborWarranty: ["Labor", "Labor Coverage", "Labor Service"]

REF_Laptop: ["Laptop", "Notebook", "Portable Computer"]

REF_Licence: ["Licence"]

REF_ListPrice: ["List Price", "From", "Original Price", "Price", "Regular Price"]

REF_Material: ["Material", "Case Material", "Chasis Material"]

REF_MaxHeight: ["Max", "Max Height", "Rear", "R"]

REF_MaxSeekTime: ["Max Seek Time", "Maximum Seek Time"]

REF_MaxStorageCapacity: ["Max Capacity", "Max Storage Capacity", "Up to", "Max"]

REF_Media: ["media"]

REF_MediaAdapter: ["Bridge Media Adapter", "Digital Media Reader", "Media Adapter", "Media Slot", "Media Slot", "Media Reader", "Memory Media", "Media Card Reader", "Multimedia Card Reader", "Media", "Slot", "Media Port"]

REF_MediaSoftware: ["Media", "Player"]

REF_Memory: ["Memory", "Main Memory", "System Memory"]

REF_MemorySlots: ["Slots", "Memory Slots", "Main memory slots", "Accesible memory slots"]

REF_Memory_Installed: ["Installed", "Installed Memory", "From", "Memory Size", "Configured with", "Capacity"]

REF_Memory_MaxInstallable: ["to", "up to", "upgrade to", "Upgreadable to", "Maximum", "Max", "Supported", "Max Supported", "Maximum Supported", "Max Memory", "Maximum Memory", "Max Capacity", "Maximum Capacity"]

REF_Memory_Technology: ["Memory Technology", "Type"]

REF_MinHeight: ["Min", "Min Height", "Front", "F", "thin at front", "as thin as"]

REF_MinSeekTime: ["Min Seek Time", "Minimum Seek Time"]

REF_Model: ["Model", "Model Number"]

REF_Modem: ["Modem", "Fax/Modem", "Fax Modem"]

REF_Modem_Certification: ["Modem Certification", "Telephone Certification"]

REF_Modem_Jack: ["Modem Port", "Modem Jack", "Telephone Plug"]

REF_Modem_Version: ["Version", "Speed", "Compliant"]

REF_ModuleTransferRate: ["Bandwidth", "Transfer Rate", "Module Name", "Transfer Capacity", "Speed"]

REF_MonthlyPrice: ["/mo", "/month", "As low as", "month", "a month", "monthly payments", "Payments"]

REF_MotionSensor: ["Motion Sensor", "Impact Protection", "Shock Protection", "Protection"]

REF_NamedResolution: ["Resolution"]

REF_NetPrice: ["Net Price", "Your Price", "After Rebate", "From"]

REF_NetworkAdapter: ["Network Adapter", "Communications", "Network Connection"]

REF_NoPaymentsTime: ["no payments"]

REF_NumberOfKeys: ["key", "keys"]

REF_NumberOfPorts: ["port", "jack", "ports", "jacks"]

REF_OnSiteWarranty: ["OnSite", "On Site", "On-Site"]

REF_OnlineSupport: ["OnLine", "Online", "On line", "On-line", "Chat"]

REF_OperatingSystem: ["Operating System", "Operative System", "SO", "S.O."]

REF_OpticalDrive: ["Optical Drive", "Drive", "Removable Media", "Multimedia Drive", "Fixed Optical Disk Drive"]

REF_OpticalFormat: ["Format", "Formats"]

REF_PCCardSlots: ["PCCard", "PC-Card", "PC Card", "CardBus", "PCMCIA", "Card Slot"]

REF_PCCardSlots_Type: ["Type"]

REF_PCExpressSlots: ["PCExpress", "PC-Express", "PC Express", "ExpressCard"]

REF_PCExpressSlots_Type: ["Type", "FormFactor"]

REF_PCISlots: ["PCI", "PCI connector", "PCI slots", "PCI Port", "PCI Expansion"]

REF_PartNumber: ["Part No.", "Part", "Part Number", "Product Number"]

REF_PartsWarranty: ["Parts", "Hardware", "Hardware Parts"]

REF_PhotoSoftware: ["Photo", "Photograph", "Image", "Picture"]

REF_PhysicalDescription: ["Physical Description", "Physical Specifications"]

REF_PhysicalSecurity: ["Physical Security"]

REF_PixelPitch: ["Pixel Pitch", "Pitch"]

REF_PixelResolution: ["Resolution", "Native resolution"]

REF_PointingDevice: ["Pointing Device", "Mouse", "Mouse Device"]

REF_PowerRequirements: ["Power Requirements", "Power Consumption"]

REF_Processor: ["Processor", "Microprocessor"]

REF_ProductivitySoftware: ["Productivity", "Office Productivity"]

REF_ReadSpeed: ["Read", "Max", "X", "x"]

REF_Rebate: ["Rebate", "Instant Rebate", "Instant savings", "Instantly", "off", "Price Drop", "Save"]

REF_RechargeTime: ["Recharge", "Recharge Time"]

REF_RechargeTimeOff: ["on", "On"]

REF_RechargeTimeOn: ["off", "Off"]

REF_Region: ["Region", "Zone"]

REF_Resolution: ["Resolution"]

REF_Sampling: ["Resolution", "Sampling", "Sampling Rate"]

REF_SecuritySoftware: ["Security Software", "Security", "Antivirus", "Anti-virus", "Anti-spyware", "Virus Protection", "Featured", "Trace"]

REF_SecuritySpecs: ["Security"]

REF_Shipping: ["Shipping", "ships"]

REF_Shipping_Money: ["Shipping Price"]

REF_Shipping_TimeMeasurement: ["Shipping Time", "Ships in", "Usually ships in"]

REF_SlotsNumber: ["connectors", "slots", "ports"]

REF_Software: ["Software", "Included Software", "Supplied Software", "Third-party software"]

REF_SoftwareSecurity: ["Software Security"]

REF_SoftwareVersion: ["Version", "Release", "Edition", "v", "V"]

REF_Speaker: ["Speakers"]

REF_Speed: ["Speed", "Processor Speed", "Microprocessor Speed", "Clock"]

REF_Support: ["Support", "Customer Care", "Assistance", "available"]

REF_TPM: ["TPM", "Trusted Plataform Module", "TPM chip", "TPM security device", "TCG", "Trusted Computing Group"]

REF_TVTuner: ["TV Tuner", "Television", "TV", "TV & Entretainment"]

REF_Technology: ["Technology", "Type"]

REF_TelephoneSupport: ["Telephone Support", "Telephone Assistance", "toll-free", "Call"]

REF_Tolerance: ["+", "+-", "+/-"]

REF_TotalSlots: ["total", "total memory slots"]

REF_UPCCode: ["UPC", "UPC Code", "Universal Product Code"]

REF_USB: ["USB", "Universal Serial Bus", "USB Ports", "USB Slots"]

REF_USB_Version: ["compliant", "version", "ver."]

REF_UtilitySoftware: ["Utility", "Utilities", "Configuration", "Config", "Setup", "Back-Up", "Back Up", "Recovery"]

REF_Version: ["Version", "Ver", "V"]

REF_VideoAdapter: ["Video Adapter", "Graphics", "Graphics Card", "Graphics Engine", "Graphics Subsystem", "Graphics Adapter", "Video Graphics", "Graphics Processing Unit", "Video"]

REF_VideoInterfaces: ["Video Interface", "Monitor Port", "Video port", "Output port", "Video Connector", "Monitor Connector", "Interface"]

REF_VideoMemory: ["Graphics Memory", "Video Memory"]

REF_VideoMemory_Installed: ["Video Memory", "Memory", "Video RAM", "RAM", "From", "Configured with"]

REF_VideoMemory_MaxInstallable: ["to", "up to", "Upgreadable to", "Maximum", "Max", "Supported", "Max Supported", "Maximum Supported", "Max Memory", "Maximum Memory", "Max Capacity"]

REF_VideoSoftware: ["Video", "Movie"]

REF_WarrantyServices: ["Warranty", "Services", "Standard Limited Warranty", "Limited Warranty", "Support", "Coverage", "Basic Warranty"]

REF_Webcam: ["Webcam", "Web camera", "Video camera", "Videoconference", "Videoconference camera", "Videoconferencing"]

REF_WeeklySchedule: ["a week", "Weekly schedule"]

REF_Weight: ["Weight", "starting at", "starting weight", "weighing only", "Weight Starting at"]

REF_Width: ["Width", "W"]

REF_WirelessAdapter: ["WLAN", "Wireless", "Wireless Adapter", "WLAN Adapter", "Wireless LAN", "Wireless LAN Adapter", "Wireless Network Connection", "Wireless Option", "Wireless Support", "Wi-FiÂ®", "Communications", "cable free"]

REF_WriteSpeed: ["Write", "Max", "X", "x"]

REF_hasAmbientLightSensor: ["ambient light sensor"]

REF_hasAntenna: ["Antenna"]

REF_hasBlueToothAntenna: ["Bluetooth Antenna", "Antenna"]

REF_hasCableLockSlot: ["Lock Slot", "Cable Lock Slot", "Cable Lock", "Kensington", "MicroSaver", "security slot", "security lock"]

REF_hasCenterButton: ["mouse center", "left center", "center"]

REF_hasDamageCoverage: ["Damage Coverage", "Accident Coverage", "Accidental Damage Coverage"]

REF_hasECC: ["ECC", "EDAC", "EDAC protected"]

REF_hasFingerprintSensor: ["Biometric", "Fingerprint", "Sensor", "Finger Print", "Reader"]

REF_hasHeadphoneJack: ["headphone", "headphone jack", "headphone port", "headphone output port", "headphone output"]

REF_hasInfraRed: ["Infrared", "IR", "FastIR", "Remote Control Receiver", "Remote Receiver"]

REF_hasLeftButton: ["mouse left", "left button", "left"]

REF_hasLineOutJack: ["line out", "line-out", "line out jack", "line out port"]

REF_hasMicrophoneJack: ["microphone", "microphone jack", "microphone port", "microphone input", "microphone input port", "microphone in"]

REF_hasNumericKeyPad: ["Numeric Key Pad", "Numeric Keypad"]

REF_hasPS/2: ["PS/2", "mouse port", "keyboard port"]

REF_hasParallel: ["DB-25", "Parallel", "Parallel Port", "IEEE1284", "IEEE-1284", "Centronics", "Centronics port", "parallel legacy port"]

REF_hasPasswordSecurity: ["Password", "Password Security", "Password Protecction", "Power-on password"]

REF_hasPortReplicatorConnector: ["Replicator", "Docking", "Port Replicator", "Replicator", "Port Replicator Connector", "Port Replicator Slot", "Docking Station Port", "Docking Station Connector", "Docking Station Slot"]

REF_hasRemoteControl: ["Remote Control", "RC", "Mobile Remote Control", "IR Control"]

REF_hasRightButton: ["mouse right", "right button", "right"]

REF_hasS/PDIF: ["S/PDIF", "SPDIF", "IEC958 type II", "Sony/Phillips Digital Interface Format"]

REF_hasScrollBar: ["Scroll bar"]

REF_hasScrollZone: ["scroll zone", "vertical scroll", "dedicated vertical scroll", "Up/Down pad"]

REF_hasSerial: ["9-pin serial", "RS-232", "RS232", "Serial Port", "Serial", "COM ports", "DE-9", "serial legacy port"]

REF_hasSpeaker: ["Built-in speakers"]

REF_hasSupervisorPassword: ["Supervisor", "Admin", "Administrator"]

REF_hasUserPassword: ["User"]

REF_hasVoltageAutoSensing: ["AutoSensing", "auto sensing"]

REF_isElectroStaticTouchPad: ["Electro-Static"]

REF_isRechargeable: ["Rechargeable"]

REF_isRemovable: ["Removable"]

REF_isSoftwareModem: ["software", "software modem", "soft modem", "soft-modem", "soft"]

REF_isSpillResistant: ["Spill-Resistant", "Spill resistant"]

REF_isWidescreen: ["Widescreen"]

REF_numberOfFormats: ["up to", "Formats"]

REF_supportsBluRay: ["Blue-Ray", "Blue Ray"]

REF_supportsDoubleLayer: ["DL", "Double Layer"]

Region: ["US", "USA", "Canada", "WorldWide", "Word Wide", "Word-Wide"]

RevolutionsPerMinute: ["rpm", "RPM", "r/min"]

Second: ["Second", "Sec.", "s"]

SecuritySoftware_Brand: ["Symantec", "Norton", "McAfee", "LoJack", "Trend-Micro"]

SecuritySoftware_Family: ["Antivirus", "Internet Security", "Computrace"]

Slash: ["/"]

SoftwareVersion: ["introductory version", "95", "96", "97", "98", "99", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "2009", "2010", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX", "X", "XI", "XII", "XIII", "XIV", "XV"]

SpeakerID_Brand: ["Altec Lansing", "JBL", "Harman/Kardon"]

Speaker_Type: ["mono", "stereo", "dolby", "dolby virtual"]

SubVersion: ["Small Business", "Basic", "Business", "Desktop", "Embeded", "Essentials", "full", "Home", "Home", "Student", "Teacher", "Home Basic", "Premier", "Premium", "Pro", "Professional", "Runtime Enviroment", "Server", "Service Pack 1", "Service Pack 2", "SP1", "SP2", "Standard", "Std", "Suite", "Ultimate", "Light"]

TPM_Brand: ["Atmel", "Broadcom", "Sinosun", "STMricoelectronics", "Winbond"]

TVFormat: [“1080i", “720p", “NTSC", “PAL"]

TeraByte: [“TB", “TByte", “TeraByte", “Tera Byte", “T.B."]

TerabitPerSecond: [“Terabit per second", “Tbit/s", “Tbps"]

TerabytePerSecond: [“Terabyte per second", “TB/s", “TBps"]

TwoPower: [“1", “2", “4", “8", “16", “32", “64", “128", “256", “512", “1024", “2048", “4096", “8192", “16384", “32768", “65536"]

USB_Version: [“2.0", “1.1", “v2.0", “v1.1"]

UtilitySoftware_Brand: [“Roxio", “Toshiba", “HP", “Softthinks"]

UtilitySoftware_Family: [“Backup MyPC", “ConfigFree", “Game Console", “PC Recovery"]

VideoAdapterID_Brand: [“ATI", “NVIDIA", “Intel", “SiS"]

VideoAdapterID_Family: [“GeForce", “Mobility", “Radeon", “All-in-wonder", “Fire GL", “Fire MV", “GeForce", “GeForce M", “Quadro", “Quadro NVS", “Quadro FX", “GMA", “Graphics Media Accelerator"]

VideoAdapterID_Model: [“Xpress", “X200M", “200M", “X3100", “Go 6150", “Go 7200", “Go 7300", “Go 7400", “Go 7600", “Go 7600", “950"]

VideoAdapter_Slot: [“PCI", “PCI Express", “PCI Extended", “PCIe"]

VideoInterfaces: [“VGA", “VGA out", “RGB", “DVI", “DVI-I", “DVI-D", “S-Video", “S Video", “TV-out", “TV out"]

VideoMemory_Technology: [“TurboCacheâ„¢"]

VideoMemory_isDedicated: [“dedicated"]

VideoMemory_isDiscrete: [“discrete"]

VideoMemory_isShared: [“shared", “dynamically allocated shared"]

VideoSoftware_Brand: [“Ulead", “Adobe", “Microsoft", “Muvee"]

VideoSoftware_Family: [“DVD MovieFactory", “DVD Movie Factory", “Premiere", “Movie Maker", “Windows Movie Maker", “AutoProducer"]

Volt: [“V", “volt", “volts"]

Watt: [“W", “watt", “watts"]

WattHour: [“watthour", “watts hour", “Wh", “WHr"]

Webcam_Resolution: [“VGA", “640x480", “800x600"]

Week: [“week", “wk"]

WeeklySchedule: [“7x24", “24x7"]

WirelessAdapterID_Brand: [“Atheros", “Broadcom", “Intel"]

WirelessAdapterID_Model: [“PRO Wireless", “PRO/Wireless LAN", “3945BG", “3945ABG", “4311AG", “4311BG", “4965 AGN"]

WirelessAdapter_Autentication: [“802.11b/g", “802.1x", “EAP-TLS", “EAP-TTLS", “LEAP", “PEAP-GTC", “PEAP-MSCHAPv2", “Wi-Fii Protected Access", “WPA", “WPA2"]

WirelessAdapter_Version: [“802.11 a/b/g", “802.11 a/b/g/draft-n", “802.11 pre-n", “802.11a/b/g/n", “802.11a/g/n", “802.11b/g", “draft 802.11n"]

X: [“x", “X"]

Yard: [“yard", “yd"]

Year: [“year", “Yr"]

Yes: [“Yes", “Included", “Standard", “Capable", “Ready", “Installed", “PreInstalled", “Compatible", “Bundled", “BuiltIn", “Embedded", “Integrated", “Supported", “Support"]

daysWeek: [“workdays", “working days", “weekends", “business days"]

hasMicrophone: [“mic", “microphone", “built-in microphone"]

hasOnOffSwitch: [“on/off"]

Appendix C

A Laptop Labeled Data-sheet Example

This manually labeled sequence example correspond to the first reported data-sheet in table 2.2.

['HP']: Laptop→ LaptopID→ Brand

['Pavilion']: Laptop→ LaptopID→ Family

['dv9500','t']: Laptop→ LaptopID→ Model

['series']: Laptop→ LaptopID→ Family→ REF_Family

['HP']: Laptop→ LaptopID→ Brand

['Windows','Vista™']: Laptop→ Software→ OperatingSystem

['Home']: Laptop→ Software→ OperatingSystem→ SubVersion

['Premium']: Laptop→ Software→ OperatingSystem→ SubVersion

['dv9500','t']: Laptop→ LaptopID→ Model

['series']: Laptop→ LaptopID→ Family→ REF_Family

['From...','<TAB>']: Laptop→ CommercialOffer→ NetPrice→ REF_NetPrice

['$',]: Laptop→ CommercialOffer→ NetPrice→ Money→ Currency

['after','rebate']: Laptop→ CommercialOffer→ NetPrice→ REF_NetPrice

['As','low','as']: Laptop→ CommercialOffer→ MonthlyPrice→ REF_MonthlyPrice

['$',]: Laptop→ CommercialOffer→ MonthlyPrice→ Money→ Currency

['35']: Laptop→ CommercialOffer→ MonthlyPrice→ Money→ MoneyAmount→ Integer

[]: Laptop→ CommercialOffer→ MonthlyPrice→ REF_MonthlyPrice

['90']: Laptop→ CommercialOffer→ NoPaymentsTime→ TimeMeasurement→ Magnitude→ Integer

['days']: Laptop→ CommercialOffer→ NoPaymentsTime→ TimeMeasurement→ TimeUnits→ Day

['no','payments']: Laptop→ CommercialOffer→ REF_NoPaymentsTime

['Free']: Laptop→ CommercialOffer→ Shipping→ Money

['shipping']: Laptop→ CommercialOffer→ Shipping→ REF_Shipping

['$',]: Laptop→ CommercialOffer→ Shipping→ Money→ Currency

['49']: Laptop→ CommercialOffer→ Shipping→ Money→ MoneyAmount→ Integer

['$',]: Laptop→ CommercialOffer→ Rebate→ Money→ Currency

['150']: Laptop→ CommercialOffer→ Rebate→ Money→ MoneyAmount→ Integer

['instant']: Laptop→ CommercialOffer→ Rebate→ REF_Rebate

['$',]: Laptop→ CommercialOffer→ Rebate→ Money→ Currency

['50']: Laptop→ CommercialOffer→ Rebate→ Money→ MoneyAmount→ Integer

['off']: Laptop→ CommercialOffer→ Rebate→ REF_Rebate

['upgrade','to']: Laptop→ Memory→ MaxInstallable→ REF_MaxInstallable

['2']: Laptop→ Memory→ MaxInstallable→ MemorySize→ MemoryMagnitude→ Integer

['GB']: Laptop→ Memory→ MaxInstallable→ MemorySize→ MemoryUnits→ GigaByte

['memory']: Laptop→ Memory→ REF_Memory

['on']: Laptop→ Memory→ MemorySlots→ OccupiedSlots→ Digit

['models']: Laptop→ LaptopID→ Model→ REF_Model

['Free']: Laptop→ Memory→ MemorySlots→ AvailableSlots

['upgrade','to']: Laptop→ Memory→ MaxInstallable→ REF_MaxInstallable

['LightScribe']: Laptop→ OpticalDrive→ Labeling

['HP']: Laptop→ LaptopID→ Brand

['assistance']: Laptop→ WarrantyServices→ Support→ REF_Support

['Call']: Laptop→ WarrantyServices→ Support→ TelephoneSupport→ REF_TelephoneSupport

['1']: Laptop→ WarrantyServices→ Support→ TelephoneSupport→ ToolFreeNumber→ Integer

['888']: Laptop→ WarrantyServices→ Support→ TelephoneSupport→ ToolFreeNumber→ Integer

['999']: Laptop→ WarrantyServices→ Support→ TelephoneSupport→ ToolFreeNumber→ Integer

['4747']: Laptop→ WarrantyServices→ Support→ TelephoneSupport→ ToolFreeNumber→ Integer

['Chat']: Laptop→ WarrantyServices→ Support→ OnlineSupport→ REF_OnlineSupport

['Warranty']: Laptop→ WarrantyServices→ REF_WarrantyServices

['Support']: Laptop→ WarrantyServices→ REF_WarrantyServices

['Operating','System']: Laptop→ Software→ OperatingSystem→ REF_OperatingSystem

['Productivity']: Laptop→ Software→ ProductivitySoftware→ REF_ProductivitySoftware

['Software']: Laptop→ Software→ REF_Software

['Windows','Vista']: Laptop→ Software→ OperatingSystem

['Home']: Laptop→ Software→ OperatingSystem→ SubVersion

['Premium']: Laptop→ Software→ OperatingSystem→ SubVersion

['32']: Laptop→ Software→ OperatingSystem→ ProcessorArchitecture→ BusWidth→ TwoPower

['bit']: Laptop→ Software→ OperatingSystem→ ProcessorArchitecture→ BusWidth→ Bit

['Windows','Vista']: Laptop→ Software→ OperatingSystem

['Business']: Laptop→ Software→ OperatingSystem→ SubVersion

['32']: Laptop→ Software→ OperatingSystem→ ProcessorArchitecture→ BusWidth→ TwoPower

['bit']: Laptop→ Software→ OperatingSystem→ ProcessorArchitecture→ BusWidth→ Bit

['Windows','Vista']: Laptop→ Software→ OperatingSystem

['Ultimate']: Laptop→ Software→ OperatingSystem→ SubVersion

['64']: Laptop→ Software→ OperatingSystem→ ProcessorArchitecture→ BusWidth→ TwoPower

['bit']: Laptop→ Software→ OperatingSystem→ ProcessorArchitecture→ BusWidth→ Bit

['Processor']: Laptop→ Processor→ REF_Processor

['processor']: Laptop→ Processor→ REF_Processor

['Intel']: Laptop→ Processor→ ProcessorID→ Brand

['Core','(','TM',')','2','Duo']: Laptop→ Processor→ ProcessorID→ Family

['T7100']: Laptop→ Processor→ ProcessorID→ Model

['1.8']: Laptop→ Processor→ Speed→ FrequencyMeasurement→ Magnitude→ Decimal

['GHz']: Laptop→ Processor→ Speed→ FrequencyMeasurement→ FrequencyUnits→ GigaHertz

['2']: Laptop→ Processor→ Cache→ CacheSize→ MemorySize→ MemoryMagnitude→ TwoPower

['MB']: Laptop→ Processor→ Cache→ CacheSize→ MemorySize→ MemoryUnits→ MegaByte

['L2',]: Laptop→ Processor→ Cache→ CacheLevel

['Cache']: Laptop→ Processor→ Cache→ REF_Cache

['Intel']: Laptop→ Processor→ ProcessorID→ Brand

['Core','(','TM',')','2','Duo']: Laptop→ Processor→ ProcessorID→ Family

['T7300']: Laptop→ Processor→ ProcessorID→ Model

['2.0']: Laptop→ Processor→ Speed→ FrequencyMeasurement→ Magnitude→ Decimal

['GHz']: Laptop→ Processor→ Speed→ FrequencyMeasurement→ FrequencyUnits→ GigaHertz

['4']: Laptop→ Processor→ Cache→ CacheSize→ MemorySize→ MemoryMagnitude→ TwoPower

['MB']: Laptop→ Processor→ Cache→ CacheSize→ MemorySize→ MemoryUnits→ MegaByte

['L2',]: Laptop→ Processor→ Cache→ CacheLevel

['Cache']: Laptop→ Processor→ Cache→ REF_Cache

['Intel']: Laptop→ Processor→ ProcessorID→ Brand

['R']: Laptop→ PhysicalDescription→ Dimensions→ Height→ MaxHeight→ REF_MaxHeight

['Core','(','TM',')','2','Duo']: Laptop→ Processor→ ProcessorID→ Family

['T7500']: Laptop→ Processor→ ProcessorID→ Model

['2.2']: Laptop→ Processor→ Speed→ FrequencyMeasurement→ Magnitude→ Decimal

['GHz']: Laptop→ Processor→ Speed→ FrequencyMeasurement→ FrequencyUnits→ GigaHertz

['MB']: Laptop→ Processor→ Cache→ CacheSize→ MemorySize→ MemoryUnits→ MegaByte

['L2',]: Laptop→ Processor→ Cache→ CacheLevel

['Cache']: Laptop→ Processor→ Cache→ REF_Cache

['Display']: Laptop→ Display→ REF_Display

['17.0']: Laptop→ Display→ Diagonal→ DistanceMagnitude

['"']: Laptop→ Display→ Diagonal→ Inch

['WXGA','+',]: Laptop→ Display→ NamedResolution

['BrightView']: Laptop→ Display→ Technology

['Widescreen']: Laptop→ Display→ isWidescreen→ REF_isWidescreen

['1440',]: Laptop→ Display→ PixelResolution→ PixelResolutionHorizontal

['x',]: Laptop→ Display→ PixelResolution→ X

['900']: Laptop→ Display→ PixelResolution→ PixelResolutionVertical

['17.0']: Laptop→ Display→ Diagonal→ Magnitude→ Decimal

['"']: Laptop→ Display→ Diagonal→ Inch

['WSXGA','+',]: Laptop→ Display→ NamedResolution

['BrightView']: Laptop→ Display→ Technology

['Widescreen']: Laptop→ Display→ isWidescreen→ REF_isWidescreen

['1680',]: Laptop→ Display→ PixelResolution→ PixelResolutionHorizontal

['x',]: Laptop→ Display→ PixelResolution→ X

['1050']: Laptop→ Display→ PixelResolution→ PixelResolutionVertical

['Memory']: Laptop→ Memory→ REF_Memory

['memory']: Laptop→ Memory→ REF_Memory

['1']: Laptop→ Memory→ Installed→ MemorySize→ MemoryMagnitude→ Integer

['GB']: Laptop→ Memory→ Installed→ MemorySize→ MemoryUnits→ GigaByte

['DDR2']: Laptop→ Memory→ Technology

['System','Memory']: Laptop→ Memory→ REF_Memory

['2']: Laptop→ Memory→ MemorySlots→ OccupiedSlots→ Digit

['Dimm']: Laptop→ Memory→ ModuleType

['$',]: Laptop→ CommercialOffer→ Rebate→ Money→ Currency

['50']: Laptop→ CommercialOffer→ Rebate→ Money→ MoneyAmount→ Integer

['off']: Laptop→ CommercialOffer→ Rebate→ REF_Rebate

['from']: Laptop→ Memory→ Installed→ REF_Installed

['1']: Laptop→ Memory→ Installed→ MemorySize→ MemoryMagnitude→ Integer

['GB']: Laptop→ Memory→ Installed→ MemorySize→ MemoryUnits→ GigaByte

['2']: Laptop→ Memory→ MemorySlots→ OccupiedSlots→ Digit

['Dimm']: Laptop→ Memory→ ModuleType

['to']: Laptop→ Memory→ MaxInstallable→ REF_MaxInstallable

['2']: Laptop→ Memory→ MaxInstallable→ MemorySize→ MemoryMagnitude→ Integer

['GB']: Laptop→ Memory→ MaxInstallable→ MemorySize→ MemoryUnits→ GigaByte

['2']: Laptop→ Memory→ MemorySlots→ OccupiedSlots→ Digit

['Dimm']: Laptop→ Memory→ ModuleType

['Graphics','Card']: Laptop→ VideoAdapter→ REF_VideoAdapter

['Intel']: Laptop→ VideoAdapter→ VideoAdapterID→ Brand

['Graphics','Media','Accelerator']: Laptop→ VideoAdapter→ VideoAdapterID→ Family

['X3100']: Laptop→ VideoAdapter→ VideoAdapterID→ Model

['notebook']: Laptop→ REF_Laptop

['HP']: Laptop→ LaptopID→ Brand

['Microphone']: Laptop→ AudioAdapter→ hasMicrophone

['HP']: Laptop→ LaptopID→ Brand

['Mic']: Laptop→ AudioAdapter→ hasMicrophone

['Fingerprint','Reader']: Laptop→ SecuritySpecs→ PhysicalSecurity→ hasFingerprintSensor→ REF_hasFingerprintSensor

['HP']: Laptop→ LaptopID→ Brand

['Microphone',]: Laptop→ AudioAdapter→ hasMicrophone

['Webcam']: Laptop→ Webcam→ REF_Webcam

['HP']: Laptop→ LaptopID→ Brand

['Fingerprint','Reader',]: Laptop→ SecuritySpecs→ PhysicalSecurity→ hasFingerprintSensor→ REF_hasFingerprintSensor

['Webcam']: Laptop→ Webcam→ REF_Webcam

['Networking']: Laptop→ NetworkAdapter→ LANAdapter→ REF_LANAdapter

['network','port']: Laptop→ NetworkAdapter→ LANAdapter→ Jack→ REF_Jack

['Intel']: Laptop→ WirelessAdapter→ WirelessAdapterID→ Brand

['PRO','/','Wireless']: Laptop→ NetworkAdapter→ WirelessAdapter→ WirelessAdapterID→ Model

['4965','AGN']: Laptop→ NetworkAdapter→ WirelessAdapter→ WirelessAdapterID→ Model

['Network','Connection']: Laptop→ NetworkAdapter→ REF_NetworkAdapter

['Intel']: Laptop→ NetworkAdapter→ WirelessAdapter→ WirelessAdapterID→ Brand

['PRO','/','Wireless']: Laptop→ NetworkAdapter→ WirelessAdapter→ WirelessAdapterID→ Model

['4965','AGN']: Laptop→ NetworkAdapter→ WirelessAdapter→ WirelessAdapterID→ Model

['Network']: Laptop→ NetworkAdapter→ LANAdapter→ REF_LANAdapter

['Bluetooth']: Laptop→ NetworkAdapter→ BlueToothAdapter→ REF_BlueToothAdapter

['Broadband','Wireless']: Laptop→ NetworkAdapter→ BroadBandWirelessAdapter→ REF_BroadBandWirelessAdapter

['Verizon']: Laptop→ NetworkAdapter→ BroadBandWirelessAdapter→ BroadBandWirelessAdapterID→ Brand

['Wireless']: Laptop→ NetworkAdapter→ WirelessAdapter→ REF_WirelessAdapter

['broadband']: Laptop→ NetworkAdapter→ BroadBandWirelessAdapter→ REF_BroadBandWirelessAdapt

['network']: Laptop→ NetworkAdapter→ LANAdapter→ REF_LANAdapter

['Verizon']: Laptop→ NetworkAdapter→ BroadBandWirelessAdapter→ BroadBandWirelessAdapterID→ Brand

['Wireless']: Laptop→ NetworkAdapter→ WirelessAdapter→ REF_WirelessAdapter

['V','740']: Laptop→ NetworkAdapter→ BroadBandWirelessAdapter→ BroadBandWirelessAdapterID→ Model

['ExpressCard']: Laptop→ ExpansionSlots→ PCExpressSlots→ REF_PCExpressSlots

['Hard','Drive']: Laptop→ HardDisk→ REF_HardDisk

['hard','disk','drive']: Laptop→ HardDisk→ REF_HardDisk

['160']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryMagnitude→ Integer

['GB']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryUnits→ GigaByte

['5400',]: Laptop→ HardDisk→ HDRotationSpeed→ HDRotationSpeedMagnitude

['RPM']: Laptop→ HardDisk→ HDRotationSpeed→ RevolutionsPerMinute

['SATA']: Laptop→ HardDisk→ Controller

['Hard','Drive']: Laptop→ HardDisk→ REF_HardDisk

['240']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryMagnitude→ Integer

['GB']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryUnits→ GigaByte

['5400',]: Laptop→ HardDisk→ HDRotationSpeed→ HDRotationSpeedMagnitude

['RPM']: Laptop→ HardDisk→ HDRotationSpeed→ RevolutionsPerMinute

['SATA']: Laptop→ HardDisk→ Controller

['Hard','Drive']: Laptop→ HardDisk→ REF_HardDisk

['120']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryMagnitude→ Integer

['GB']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryUnits→ GigaByte

['320']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryMagnitude→ Integer

['GB']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryUnits→ GigaByte

['5400',]: Laptop→ HardDisk→ HDRotationSpeed→ HDRotationSpeedMagnitude

['RPM']: Laptop→ HardDisk→ HDRotationSpeed→ RevolutionsPerMinute

['SATA']: Laptop→ HardDisk→ Controller

['Hard','Drive']: Laptop→ HardDisk→ REF_HardDisk

['160']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryMagnitude→ Integer

['GB']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryUnits→ GigaByte

['400']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryMagnitude→ Integer

['GB']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryUnits→ GigaByte

['4200',]: Laptop→ HardDisk→ HDRotationSpeed→ HDRotationSpeedMagnitude

['RPM']: Laptop→ HardDisk→ HDRotationSpeed→ RevolutionsPerMinute

['SATA']: Laptop→ HardDisk→ Controller

['Hard','Drive']: Laptop→ HardDisk→ REF_HardDisk

['200']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryMagnitude→ Integer

['GB']: Laptop→ HardDisk→ Capacity→ MemorySize→ MemoryUnits→ GigaByte

['CD']: Laptop→ OpticalDrive→ OpticalFormat→ CDFormat→ REF_CDFormat

['DVD']: Laptop→ OpticalDrive→ OpticalFormat→ DVDFormat→ REF_DVDFormat

['Drive']: Laptop→ OpticalDrive→ REF_OpticalDrive

['Optical','drives']: Laptop→ OpticalDrive→ REF_OpticalDrive

['CDs']: Laptop→ OpticalDrive→ OpticalFormat→ CDFormat→ REF_CDFormat

['DVDs']: Laptop→ OpticalDrive→ OpticalFormat→ DVDFormat→ REF_DVDFormat

['8',]: Laptop→ OpticalDrive→ OpticalFormat→ DVDFormat→ ReadSpeed→ OpticalDriveSpeedMagnitude

['X']: Laptop→ OpticalDrive→ OpticalFormat→ DVDFormat→ ReadSpeed→ REF_ReadSpeed

['DVD','+','/','-','R']: Laptop→ OpticalDrive→ OpticalFormat→ DVDFormat→ Media

['Double','Layer']: Laptop→ OpticalDrive→ DVDLayer

['LightScribe']: Laptop→ OpticalDrive→ Labeling

['DVD','+','/','-','RW']: Laptop→ OpticalDrive→ OpticalFormat→ DVDFormat→ Media

['Double','Layer']: Laptop→ OpticalDrive→ DVDLayer

['TV']: Laptop→ VideoAdapter→ TVTuner→ REF_TVTuner

['TV']: Laptop→ VideoAdapter→ TVTuner→ REF_TVTuner

['No']: Laptop→ VideoAdapter→ TVTuner→ hasTVTunner→ Boolean→ Not

['TV']: Laptop→ VideoAdapter→ TVTuner→ REF_TVTuner

['remote','control']: Laptop→ InputDevice→ hasRemoteControl→ REF_hasRemoteControl

['HP']: Laptop→ LaptopID→ Brand

['Expresscard']: Laptop→ ExpansionSlots→ PCExpressSlots→ REF_PCExpressSlots

['TV','Tuner']: Laptop→ VideoAdapter→ TVTuner→ REF_TVTuner

['Windows','Vista']: Laptop→ Software→ OperatingSystem

['Notebook']: Laptop→ REF_Laptop

['Primary','Battery']: Laptop→ Battery→ REF_Battery

['battery']: Laptop→ Battery→ REF_Battery

['Notebook']: Laptop→ REF_Laptop

['8']: Laptop→ Battery→ Cells→ Digit

['Cell']: Laptop→ Battery→ Cells→ REF_Cells

['Lithium','Ion']: Laptop→ Battery→ Technology

['Battery']: Laptop→ Battery→ REF_Battery

['8']: Laptop→ Battery→ Cells→ Digit

['Cell']: Laptop→ Battery→ Cells→ REF_Cells

['Lithium','Ion']: Laptop→ Battery→ Technology

['Battery']: Laptop→ Battery→ REF_Battery

['8']: Laptop→ Battery→ Cells→ Digit

['Cell']: Laptop→ Battery→ Cells→ REF_Cells

['Lithium','Ion']: Laptop→ Battery→ Technology

['Battery']: Laptop→ Battery→ REF_Battery

['Two']: Laptop→ Battery→ Cells→ Digit

['Capacity']: Laptop→ Battery→ ElectricalCurrentCapacity→ REF_ElectricalCurrentCapacity

['8']: Laptop→ Battery→ Cells→ Digit

['Cell']: Laptop→ Battery→ Cells→ REF_Cells

['Lithium','Ion']: Laptop→ Battery→ Technology

['Batteries']: Laptop→ Battery→ REF_Battery

['Security','Software']: Laptop→ Software→ REF_Software

['Norton']: Laptop→ Software→ SecuritySoftware→ Brand

['Internet','Security']: Laptop→ Software→ SecuritySoftware→ Family

['2007']: Laptop→ Software→ SecuritySoftware→ SoftwareVersion

['15']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ Magnitude→ Integer

['Months']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ TimeUnits→ Month

['Norton']: Laptop→ Software→ SecuritySoftware→ Brand

['Internet','Security']: Laptop→ Software→ SecuritySoftware→ Family

['2007']: Laptop→ Software→ SecuritySoftware→ SoftwareVersion

['24']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ Magnitude→ Integer

['Months']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ TimeUnits→ Month

['Norton']: Laptop→ Software→ SecuritySoftware→ Brand

['Internet','Security']: Laptop→ Software→ SecuritySoftware→ Family

['2007']: Laptop→ Software→ SecuritySoftware→ SoftwareVersion

['36']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ Magnitude→ Integer

['Months']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ TimeUnits→ Month

['HP']: Laptop→ LaptopID→ Brand

['Featured']: Laptop→ Software→ SecuritySoftware→ REF_SecuritySoftware

['Software']: Laptop→ Software→ REF_Software

['Computrace']: Laptop→ Software→ SecuritySoftware→ Family

['LoJack']: Laptop→ Software→ SecuritySoftware→ Brand

['Software']: Laptop→ Software→ REF_Software

['Computrace']: Laptop→ Software→ SecuritySoftware→ Family

['LoJack']: Laptop→ Software→ SecuritySoftware→ Brand

['Laptops']: Laptop→ REF_Laptop

['One']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ Magnitude→ Digit

['Year']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ TimeUnits→ Year

['Computrace']: Laptop→ Software→ SecuritySoftware→ Family

['LoJack']: Laptop→ Software→ SecuritySoftware→ Brand

['Laptops']: Laptop→ REF_Laptop

['Three']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ Magnitude→ Digit

['Years']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ TimeUnits→ Year

['Computrace']: Laptop→ Software→ SecuritySoftware→ Family

['LoJack']: Laptop→ Software→ SecuritySoftware→ Brand

['Laptops']: Laptop→ REF_Laptop

['Four']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ Magnitude→ Digit

['Years']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ TimeUnits→ Year

['Back','-','Up']: Laptop→ Software→ UtilitySoftware→ REF_UtilitySoftware

['Utilities']: Laptop→ Software→ UtilitySoftware→ REF_UtilitySoftware

['Software']: Laptop→ Software→ REF_Software

['Roxio']: Laptop→ Software→ UtilitySoftware→ Brand

['Backup','MyPC']: Laptop→ Software→ UtilitySoftware→ Family

['Recovery']: Laptop→ Software→ UtilitySoftware→ REF_UtilitySoftware

['Media']: Laptop→ Software→ MediaSoftware→ REF_MediaSoftware

['Recovery']: Laptop→ Software→ UtilitySoftware→ REF_UtilitySoftware

['DVD']: Laptop→ OpticalDrive→ OpticalFormat→ DVDFormat→ REF_DVDFormat

['Windows','Vista']: Laptop→ Software→ OperatingSystem

['Home']: Laptop→ Software→ OperatingSystem→ SubVersion

['Premium']: Laptop→ Software→ OperatingSystem→ SubVersion

['Recovery']: Laptop→ Software→ UtilitySoftware→ REF_UtilitySoftware

['DVD']: Laptop→ OpticalDrive→ OpticalFormat→ DVDFormat→ REF_DVDFormat

['Windows','Vista']: Laptop→ Software→ OperatingSystem

['Ultimate']: Laptop→ Software→ OperatingSystem→ SubVersion

['Recovery']: Laptop→ Software→ UtilitySoftware→ REF_UtilitySoftware

['DVD']: Laptop→ OpticalDrive→ OpticalFormat→ DVDFormat→ REF_DVDFormat

['Windows','Vista']: Laptop→ Software→ OperatingSystem

['Business']: Laptop→ Software→ OperatingSystem→ SubVersion

['Productivity']: Laptop→ Software→ ProductivitySoftware→ REF_ProductivitySoftware

['Software']: Laptop→ Software→ REF_Software

['Microsoft']: Laptop→ Software→ ProductivitySoftware→ Brand

['Works']: Laptop→ Software→ ProductivitySoftware→ Family

['8.0']: Laptop→ Software→ ProductivitySoftware→ SoftwareVersion→ Decimal

['Corel']: Laptop→ Software→ ProductivitySoftware→ Brand

['WordPerfect']: Laptop→ Software→ ProductivitySoftware→ Family

['Office','X3']: Laptop→ Software→ ProductivitySoftware→ Family

['Microsoft']: Laptop→ Software→ ProductivitySoftware→ Brand

['Works']: Laptop→ Software→ ProductivitySoftware→ Family

['Suite']: Laptop→ Software→ ProductivitySoftware→ SubVersion

['2006']: Laptop→ Software→ ProductivitySoftware→ SoftwareVersion

['Word']: Laptop→ Software→ ProductivitySoftware→ Family

['Microsoft']: Laptop→ Software→ ProductivitySoftware→ Brand

['Office']: Laptop→ Software→ ProductivitySoftware→ Family

['Basic']: Laptop→ Software→ ProductivitySoftware→ SubVersion

['2007']: Laptop→ Software→ ProductivitySoftware→ SoftwareVersion

['Microsoft']: Laptop→ Software→ ProductivitySoftware→ Brand

['Office']: Laptop→ Software→ ProductivitySoftware→ Family

['Home']: Laptop→ Software→ ProductivitySoftware→ SubVersion

['Student']: Laptop→ Software→ ProductivitySoftware→ SubVersion

['2007']: Laptop→ Software→ ProductivitySoftware→ SoftwareVersion

['Microsoft']: Laptop→ Software→ ProductivitySoftware→ Brand

['Office']: Laptop→ Software→ ProductivitySoftware→ Family

['Small','Business']: Laptop→ Software→ ProductivitySoftware→ SubVersion

['2007']: Laptop→ Software→ ProductivitySoftware→ SoftwareVersion

['Microsoft']: Laptop→ Software→ ProductivitySoftware→ Brand

['Office']: Laptop→ Software→ ProductivitySoftware→ Family

['Professional']: Laptop→ Software→ ProductivitySoftware→ SubVersion

['2007']: Laptop→ Software→ ProductivitySoftware→ SoftwareVersion

['Premium']: Laptop→ Software→ ProductivitySoftware→ SubVersion

['Photography']: Laptop→ Software→ PhotoSoftware→ REF_PhotoSoftware

['Software']: Laptop→ Software→ REF_Software

['Software']: Laptop→ Software→ REF_Software

['Corel']: Laptop→ Software→ PhotoSoftware→ Brand

['Photo','Album']: Laptop→ Software→ PhotoSoftware→ Family

['6']: Laptop→ Software→ PhotoSoftware→ SoftwareVersion

['Corel']: Laptop→ Software→ ProductivitySoftware→ Brand

['Paint','Shop']: Laptop→ Software→ PhotoSoftware→ Family

['Pro']: Laptop→ Software→ PhotoSoftware→ SubVersion

['Photo']: Laptop→ Software→ PhotoSoftware→ REF_PhotoSoftware

['Corel']: Laptop→ Software→ PhotoSoftware→ Brand

['Photo','Album']: Laptop→ Software→ PhotoSoftware→ Family

['6']: Laptop→ Software→ PhotoSoftware→ SoftwareVersion

['Paint','Shop']: Laptop→ Software→ PhotoSoftware→ Family

['Pro']: Laptop→ Software→ PhotoSoftware→ SubVersion

['X1']: Laptop→ Software→ PhotoSoftware→ SoftwareVersion

['Burner']: Laptop→ Software→ BurningSoftware→ Family

['Software']: Laptop→ Software→ REF_Software

['Roxio']: Laptop→ Software→ BurningSoftware→ Brand

['Creator']: Laptop→ Software→ BurningSoftware→ Family

['V']: Laptop→ Software→ BurningSoftware→ SoftwareVersion

['9']: Laptop→ Software→ BurningSoftware→ SoftwareVersion→ Integer

['Roxio']: Laptop→ Software→ BurningSoftware→ Brand

['Creator']: Laptop→ Software→ BurningSoftware→ Family

['Premier']: Laptop→ Software→ BurningSoftware→ SubVersion

['V']: Laptop→ Software→ BurningSoftware→ SoftwareVersion

['9']: Laptop→ Software→ BurningSoftware→ SoftwareVersion→ Integer

['Display','<TAB>"<TAB>']: Laptop→ Display→ REF_Display

['17.0']: Laptop→ Display→ Diagonal→ DistanceMagnitude

['"']: Laptop→ Display→ Diagonal→ Inch

['WXGA','+',]: Laptop→ Display→ NamedResolution

['BrightView']: Laptop→ Display→ Technology

['Widescreen']: Laptop→ Display→ isWidescreen→ REF_isWidescreen

['Display']: Laptop→ Display→ REF_Display

['WSXGA','+',]: Laptop→ Display→ NamedResolution

['BrightView']: Laptop→ Display→ Technology

['Dimensions','<TAB>"<TAB>']: Laptop→ PhysicalDescription→ Dimensions→ REF_Dimensions

['15.16']: Laptop→ PhysicalDescription→ Dimensions→ Depth→ DistanceMeasurement→ Magnitude→ Decimal

['"']: Laptop→ PhysicalDescription→ Dimensions→ Depth→ DistanceMeasurement→ DistanceUnits→ Inch

['L']: Laptop→ PhysicalDescription→ Dimensions→ Depth→ REF_Depth

['x']: Laptop→ PhysicalDescription→ Dimensions→ X

['11.65']: Laptop→ PhysicalDescription→ Dimensions→ Width→ DistanceMeasurement→ Magnitude→ Decimal

['"']: Laptop→ PhysicalDescription→ Dimensions→ Width→ DistanceMeasurement→ DistanceUnits→ Inch

['W']: Laptop→ PhysicalDescription→ Dimensions→ Width→ REF_Width

['x']: Laptop→ PhysicalDescription→ Dimensions→ X

['1.57']: Laptop→ PhysicalDescription→ Dimensions→ Height→ DistanceMeasurement→ Magnitude→ Decimal

['"']: Laptop→ PhysicalDescription→ Dimensions→ Height→ DistanceMeasurement→ DistanceUnits→ Inch

['Weight','<TAB>"<TAB>']: Laptop→ PhysicalDescription→ Weight→ REF_Weight

['7.7']: Laptop→ PhysicalDescription→ Weight→ WeightMeasurement→ Magnitude→ Decimal

['lbs']: Laptop→ PhysicalDescription→ Weight→ WeightMeasurement→ WeightUnits→ Pound

['Weight']: Laptop→ PhysicalDescription→ Weight→ REF_Weight

['configuration']: Laptop→ Software→ UtilitySoftware→ REF_UtilitySoftware

['Communications','<TAB>"<TAB>']: Laptop→ NetworkAdapter→ REF_NetworkAdapter

['Intel']: Laptop→ NetworkAdapter→ WirelessAdapter→ WirelessAdapterID→ Brand

['PRO','/','Wireless']: Laptop→ NetworkAdapter→ WirelessAdapter→ WirelessAdapterID→ Model

['4965','AGN']: Laptop→ NetworkAdapter→ WirelessAdapter→ WirelessAdapterID→ Model

['Network']: Laptop→ NetworkAdapter→ LANAdapter→ REF_LANAdapter

['optional']: Laptop→ NetworkAdapter→ BlueToothAdapter→ hasBlueToothAdapter→ Boolean→ Not

['Bluetooth']: Laptop→ NetworkAdapter→ BlueToothAdapter→ REF_BlueToothAdapter

['Broadband','Wireless']: Laptop→ NetworkAdapter→ BroadBandWirelessAdapter→ REF_BroadBandWirelessAdapter

['Verizon']: Laptop→ NetworkAdapter→ BroadBandWirelessAdapter→ BroadBandWirelessAdapterID→ Brand

['Wireless']: Laptop→ NetworkAdapter→ WirelessAdapter→ REF_WirelessAdapter

['V','740']: Laptop→ NetworkAdapter→ BroadBandWirelessAdapter→ BroadBandWirelessAdapterID→ Model

['ExpressCard']: Laptop→ ExpansionSlots→ PCExpressSlots→ REF_PCExpressSlots

['Graphics','<TAB>"<TAB>']: Laptop→ VideoAdapter→ REF_VideoAdapter

['Intel']: Laptop→ VideoAdapter→ VideoAdapterID→ Brand

['Graphics','Media','Accelerator']: Laptop→ VideoAdapter→ VideoAdapterID→ Family

['X3100']: Laptop→ VideoAdapter→ VideoAdapterID→ Model

['PCI','expansion']: Laptop→ ExpansionSlots→ PCISlots→ REF_PCISlots

['Expansion','port']: Laptop→ ExpansionSlots→ REF_ExpansionSlots

['3']: Laptop→ ExpansionSlots→ PCISlots→ SlotsNumber→ Digit

['connector']: Laptop→ ExpansionSlots→ PCISlots→ SlotsNumber→ REF_SlotsNumber

['Memory','<TAB>"<TAB>']: Laptop→ Memory→ REF_Memory

['From']: Laptop→ Memory→ Installed→ REF_Installed

['1.0']: Laptop→ Memory→ Installed→ MemorySize→ MemoryMagnitude→ Decimal

['GB']: Laptop→ Memory→ Installed→ MemorySize→ MemoryUnits→ GigaByte

['to']: Laptop→ Memory→ MaxInstallable→ REF_MaxInstallable

['2.0']: Laptop→ Memory→ MaxInstallable→ MemorySize→ MemoryMagnitude→ Decimal

['GB']: Laptop→ Memory→ MaxInstallable→ MemorySize→ MemoryUnits→ GigaByte

['DDR2']: Laptop→ Memory→ Technology

['SDRAM']: Laptop→ Memory→ Technology

['Total','memory','slots','<TAB>"<TAB>']: Laptop→ Memory→ MemorySlots→ TotalSlots→ REF_TotalSlots

['2']: Laptop→ Memory→ MemorySlots→ TotalSlots→ Digit

['DIMM']: Laptop→ Memory→ ModuleType

['Maximum','memory']: Laptop→ Memory→ MaxInstallable→ REF_MaxInstallable

['2']: Laptop→ Memory→ MaxInstallable→ MemorySize→ MemoryMagnitude→ Integer

['GB']: Laptop→ Memory→ MaxInstallable→ MemorySize→ MemoryUnits→ GigaByte

['Hard','disk','drive']: Laptop→ HardDisk→ REF_HardDisk

['s']: Laptop→ HardDisk→ MinSeekTime→ TimeMeasurement→ TimeUnits→ Second

['Up','to']: Laptop→ HardDisk→ MaxStorageCapacity→ REF_MaxStorageCapacity

['400']: Laptop→ HardDisk→ MaxStorageCapacity→ MemorySize→ MemoryMagnitude→ Integer

['GB§']: Laptop→ HardDisk→ MaxStorageCapacity→ MemorySize→ MemoryUnits→ GigaByte

['4200']: Laptop→ HardDisk→ HDRotationSpeed→ HDRotationSpeedMagnitude

['rpm']: Laptop→ HardDisk→ HDRotationSpeed→ RevolutionsPerMinute

['Serial','ATA']: Laptop→ HardDisk→ Controller

['hard','drive']: Laptop→ HardDisk→ REF_HardDisk

['Primary','battery','<TAB>"<TAB>']: Laptop→ Battery→ REF_Battery

['8']: Laptop→ Battery→ Cells→ Digit

['cell']: Laptop→ Battery→ Cells→ REF_Cells

['Lithium','Ion']: Laptop→ Battery→ Technology

['battery']: Laptop→ Battery→ REF_Battery

['Battery']: Laptop→ Battery→ REF_Battery

['Front','-','side','bus']: Laptop→ Processor→ FrontSideBus→ REF_FrontSideBus

['processor']: Laptop→ Processor→ REF_Processor

['Up','to']: Laptop→ Processor→ FrontSideBus→ REF_FrontSideBus

['800']: Laptop→ Processor→ FrontSideBus→ FrequencyMagnitude

['MHz']: Laptop→ Processor→ FrontSideBus→ MegaHertz

['AC','adapter','<TAB>"<TAB>']: Laptop→ ACAdapter→ REF_ACAdapter

['65']: Laptop→ ACAdapter→ ACInput→ PowerRequirements→ PowerMeasurement→ Magnitude→ Integer

['W']: Laptop→ ACAdapter→ ACInput→ PowerRequirements→ PowerMeasurement→ PowerUnits→ Watt

['Expansion','slots','<TAB>"<TAB>']: Laptop→ ExpansionSlots→ REF_ExpansionSlots

['1']: Laptop→ ExpansionSlots→ PCExpressSlots→ SlotsNumber→ Digit

['ExpressCard','/','54']: Laptop→ ExpansionSlots→ PCExpressSlots→ Type

['Slot']: Laptop→ ExpansionSlots→ REF_ExpansionSlots

['ExpressCard','/','34']: Laptop→ ExpansionSlots→ PCExpressSlots→ Type

['ports','<TAB>"<TAB>']: Laptop→ Interfaces→ REF_Interfaces

['3']: Laptop→ Interfaces→ USB→ NumberOfPorts

['Universal','Serial','Bus']: Laptop→ Interfaces→ USB→ REF_USB

['USB']: Laptop→ Interfaces→ USB→ REF_USB

['2.0']: Laptop→ Interfaces→ USB→ Version

['IEEE','1394']: Laptop→ Interfaces→ IEEE1394→ REF_IEEE1394

['Firewire']: Laptop→ Interfaces→ IEEE1394→ REF_IEEE1394

['expansion','port']: Laptop→ ExpansionSlots→ REF_ExpansionSlots

['3']: Laptop→ ExpansionSlots→ PCISlots→ SlotsNumber→ Digit

['TV','out']: Laptop→ VideoAdapter→ VideoInterfaces

['S','-','video']: Laptop→ VideoAdapter→ VideoInterfaces

['Integrated']: Laptop→ Interfaces→ hasInfraRed→ Boolean→ Yes

['IR']: Laptop→ Interfaces→ hasInfraRed→ REF_hasInfraRed

['remote','control','receiver']: Laptop→ Interfaces→ hasInfraRed→ REF_hasInfraRed

['5']: Laptop→ MediaAdapter→ NumberOfMediaFormats→ Integer

['in','-','1']: Laptop→ MediaAdapter→ NumberOfMediaFormats

['media','card','reader']: Laptop→ MediaAdapter→ REF_MediaAdapter

['microphone']: Laptop→ AudioAdapter→ hasMicrophone

['RJ','-','11']: Laptop→ Modem→ Jack

['modem']: Laptop→ Modem→ REF_Modem

['RJ','-','45']: Laptop→ NetworkAdapter→ LANAdapter→ Jack

['LAN']: Laptop→ NetworkAdapter→ LANAdapter→ REF_LANAdapter

['VGA']: Laptop→ VideoAdapter→ VideoInterfaces

['Speakers','<TAB>"<TAB>']: Laptop→ AudioAdapter→ Speaker→ REF_Speaker

['Integrated']: Laptop→ AudioAdapter→ Speaker→ hasSpeaker→ Boolean→ Yes

['Altec','Lansing']: Laptop→ AudioAdapter→ Speaker→ SpeakerID→ Brand

['stereo']: Laptop→ AudioAdapter→ Speaker→ Type

['speakers']: Laptop→ AudioAdapter→ Speaker→ REF_Speaker

['Software']: Laptop→ Software→ REF_Software

['HP']: Laptop→ Software→ PhotoSoftware→ Brand

['PhotoSmart']: Laptop→ Software→ PhotoSoftware→ Family

['Essentials']: Laptop→ Software→ PhotoSoftware→ SubVersion

['RealRhapsody']: Laptop→ Software→ MediaSoftware→ Family

['Muvee']: Laptop→ Software→ MediaSoftware→ Brand

['AutoProducer']: Laptop→ Software→ MediaSoftware→ Family

['Basic']: Laptop→ Software→ MediaSoftware→ SubVersion

['Edition']: Laptop→ Software→ MediaSoftware→ SoftwareVersion→ REF_SoftwareVersion

['5']: Laptop→ Software→ MediaSoftware→ SoftwareVersion→ Integer

['20']: Laptop→ Software→ MediaSoftware→ Licence→ TrialPeriod→ TimeMeasurement→ Magnitude→ Integer

['day']: Laptop→ Software→ MediaSoftware→ Licence→ TrialPeriod→ TimeMeasurement→ TimeUnits→ Day

['trial']: Laptop→ Software→ MediaSoftware→ Licence→ LicenceType

['full']: Laptop→ Software→ MediaSoftware→ Licence→ LicenceType

['version']: Laptop→ Software→ MediaSoftware→ SoftwareVersion→ REF_SoftwareVersion

['Adobe']: Laptop→ Software→ ProductivitySoftware→ Brand

['Acrobat','Reader']: Laptop→ Software→ ProductivitySoftware→ Family

['Microsoft']: Laptop→ Software→ ProductivitySoftware→ Brand

['Works']: Laptop→ Software→ ProductivitySoftware→ Family

['8.0']: Laptop→ Software→ ProductivitySoftware→ SoftwareVersion→ Decimal

['Microsoft']: Laptop→ Software→ MediaSoftware→ Brand

['Windows','Media','Player']: Laptop→ Software→ MediaSoftware→ Family

['11']: Laptop→ Software→ MediaSoftware→ SoftwareVersion→ Integer

['HP']: Laptop→ Software→ PhotoSoftware→ Brand

['Roxio']: Laptop→ Software→ BurningSoftware→ Brand

['Creator']: Laptop→ Software→ BurningSoftware→ Family

['9']: Laptop→ Software→ BurningSoftware→ SoftwareVersion→ Integer

['Basic']: Laptop→ Software→ BurningSoftware→ SubVersion

['HP']: Laptop→ Software→ MediaSoftware→ Brand

['QuickPlay']: Laptop→ Software→ MediaSoftware→ Family

['Software']: Laptop→ Software→ REF_Software

['introductory','versions','<TAB>"<TAB>']: Laptop→ Software→ SecuritySoftware→ SoftwareVersion

['Symantec']: Laptop→ Software→ SecuritySoftware→ Brand

['Norton']: Laptop→ Software→ SecuritySoftware→ Brand

['Internet','Security']: Laptop→ Software→ SecuritySoftware→ Family

['2007']: Laptop→ Software→ SecuritySoftware→ SoftwareVersion

['60']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ Magnitude→ Integer

['days']: Laptop→ Software→ SecuritySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ TimeUnits→ Day

['30']: Laptop→ CommercialOffer→ Shipping→ TimeMeasurement→ Magnitude→ Integer

['day']: Laptop→ CommercialOffer→ Shipping→ TimeMeasurement→ TimeUnits→ Day

['free']: Laptop→ CommercialOffer→ Shipping→ Money

['trial']: Laptop→ Software→ ProductivitySoftware→ Licence→ LicenceType

['Microsoft']: Laptop→ Software→ ProductivitySoftware→ Brand

['Office']: Laptop→ Software→ ProductivitySoftware→ Family

['Home']: Laptop→ Software→ ProductivitySoftware→ SubVersion

['Student']: Laptop→ Software→ ProductivitySoftware→ SubVersion

['2007']: Laptop→ Software→ ProductivitySoftware→ SoftwareVersion

['Edition']: Laptop→ Software→ ProductivitySoftware→ SoftwareVersion→ REF_SoftwareVersion

['60']: Laptop→ Software→ ProductivitySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ Magnitude→ Integer

['day']: Laptop→ Software→ ProductivitySoftware→ Licence→ TrialPeriod→ TimeMeasurement→ TimeUnits→ Day

['trial']: Laptop→ Software→ ProductivitySoftware→ Licence→ LicenceType

['Basic','warranty']: Laptop→ WarrantyServices→ REF_WarrantyServices

['One']: Laptop→ WarrantyServices→ Warranty→ PartsWarranty→ TimeMeasurement→ Magnitude→ Digit

['year']: Laptop→ WarrantyServices→ Warranty→ PartsWarranty→ TimeMeasurement→ TimeUnits→ Year

['hardware','parts']: Laptop→ WarrantyServices→ Warranty→ PartsWarranty→ REF_PartsWarranty

['labor','coverage']: Laptop→ WarrantyServices→ Warranty→ LaborWarranty→ REF_LaborWarranty

['One']: Laptop→ WarrantyServices→ Warranty→ LaborWarranty→ TimeMeasurement→ Magnitude→ Digit

['year']: Laptop→ WarrantyServices→ Warranty→ LaborWarranty→ TimeMeasurement→ TimeUnits→ Year

['toll','-','free']: Laptop→ WarrantyServices→ Support→ TelephoneSupport→ REF_TelephoneSupport

['24','x','7']: Laptop→ WarrantyServices→ Support→ TelephoneSupport→ WeeklySchedule

['support']: Laptop→ WarrantyServices→ Support→ REF_Support

Bibliography

[1] Eythan Adar. S-rad a simple and robust abbreviation dictionary. Technical report, HP Laboratories Technical Report, September 2002.

[2] Eneko Agirre and German Rigau. A proposal for word sense disambiguation using conceptual distance. In *Proceedings of RANLP 96*, 1996.

[3] Ergin Altintas, Elif Karsligil, and Vedat Coskun. A new semantic similarity measure evaluated in word sense disambiguation. In *Proceedings of the 15th Nordic Conference of Computational Linguistics NODALIDA 2005*, 2005.

[4] Alberto Apostolico and Concettina Guerra. The longest common subsequence problem revisited. *Algorithmica*, 2(1):315–336, 1987.

[5] B. De Baets and H. De Meyer. Transitivity-preserving fuzzification schemes for cardinality-based similarity measures. *European Journal of Operational Research 160 (2005) 726-740*, 160:726–740, 2005.

[6] Ricardo Baeza-Yates and Berthier Ribero-Neto. *Modern Information Retrieval*. Addison Wesley / ACM Press, 1999.

[7] Satanjeev Banerjee and Ted Pedersen. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Computational linguistics and intelligent text processing CICLing*, 2002.

[8] Satanjeev Banerjee and Ted Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the 18th IJCAI*, 2003.

[9] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. String mathcing with metric trees using an approximate distance. In *Proceedings of SPIRE, LNCS 2476, Lissbon, Portugal.*, 2002.

[10] T.C. Bell, J.G. Cleary, and I.H. Witten. *Text Compression*. Prentice Hall, Englewood Cliffs, 1990.

[11] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 2001.

[12] Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003.

[13] Sergey Brin. Extracting patterns and relations from the world wide web. In *WebDB Workshop at EDBT 98*, 1998.

[14] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[15] Alexander Budanitsky and Graeme Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics*, 2001.

[16] Paul Buitelaar, Philipp Cimiano, Stefania Racioppa, and Melanie Siegel. Ontology-based information extraction with soba. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2006.

[17] Chia-Hui Chang, Mohammed Kayed, Moheb Ramzy Girgis, and Khaled Shaalan. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18:1411–1428, 2006.

[18] Jeffrey T. Chang, Hinrich Schütze, and Russ B. Altman. Creating an online dictionary of abbreviations from medline. *Journal of the American Medical Informatics Association*, 9(6):612–620, 2002.

[19] Surajit Chaudhuri, Kris Ganjam, Venkatesh Ganti, and Rajeev Motwani. Robust and efficient fuzzy match for online data cleaning. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003.

[20] Boris Chidlovskii, Jon Ragetli, and Maarten de Rijke. Wrapper generation via grammar induction. In *Proceedings of the European Conference on Machine Learning*, 2000.

[21] Peter Christen. A comparison of personal name matching: Techniques and practical issues. Technical report, The Australian National University, Department of Computer Science, Faculty of Engineering and Information Technology, 2006.

[22] Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51:1523–1545, 2005.

[23] Philipp Cimiano, Siegfried Handschuh, and Steffen Staab. Towards the self-annotating web. In *Proceedings of the 13th conference on World Wide Web, 2004*, 2004.

[24] Fabio Ciravegna, Sam Chapman, Alexiei Dingli, and Yorick Wilks. Learning to harvest information for the semantic web. In *Proceedings of the 1st European Semantic Web Symposium*, Heraklion, Greece, 2004.

[25] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, 2003.

[26] Jim Cowie, Joe Guthrie, and Louise Guthrie. Lexical disambiguation using simulated annealing. In *Proceedings of COLING-92*, 1992.

[27] Hamish Cunningham. Information extraction, automatic. *Encyclopedia of Language and Linguistics, 2nd Edition*, 2005.

[28] Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, March 1964.

[29] Debabrata Dey and Sumit Sarkar. A distance-bases approach to entity reconciliation in heterogeneous databases. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):567–582, 2002.

[30] Edgar W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[31] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K.S. McCurley, S. Rajagopalan, A. Tomkins, J.A. Tomlin, and J.Y. Zienberer. A case for automated large scale semantic annotation. *Web Semantics*, 1(1), 2003.

[32] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, Jaunary 2007.

[33] David W. Embley. Toward semantic understanding an approach based on information extraction ontologies. In *Proceedings of the fifteenth Australasian database conference Dunedin, New Zealand*, volume 27, pages 3–12, 2004.

[34] Oren Etzioni, Michele Banko, and Michael J. Cafarella. Machine reading. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, 2006.

[35] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall. In *Proceedings of the 13th international conference on World Wide Web*, 2004.

[36] Dayne Freitag and Andrew McCallum. Information extraction with hmm structures learned by stochastic optimization. In *AAAI*, 2000.

[37] Dayne Freitag and Andrew Kachites McCallum. Information extraction with hmms and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Informatino Extraction*, 1999.

[38] William A. Gale, Kenneth W. Church, and David Yarowsky. One sense per discourse. In *Proceedings of the DARPA workshop on Speech and Natural Language*, 1992.

[39] Alexander Gelbukh, Grigori Sidorov, and San-Yong Han. Evolutionary approach to natural language word sense disambiguation through global coherence optimization. *WSEAS Transactions on Communications*, 2(1):11–19, January 2003 2003.

[40] Osamu Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708, 1982.

[41] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Formal Analysis in Conceptual Analysis and Knowledge Representation, Kluwer*, 1993.

[42] Siegfried Handschuh, Steffen Staab, and Fabio Ciravegna. S-cream: Semi-automatic creation of metadata. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, 2473:165–184, 2002.

[43] R. Hwang, D. Richards, and P. Winter. The steiner tree problem. *Annals of Discrete Mathematics*, 53, 1992.

[44] Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference on Research in Computational Linguistics, Taiwan*, 1997.

[45] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[46] H. Keskustalo, A. Pirkola, K. Visala, and E. Leppanen. Non-adjacent digrams improve matching of cross-lingual spelling variants. In *Proceedings of the 10th SPIRE Manaus, Brazil*, 2003.

[47] A. Kilgarriff and J. Rosenzweig. Framework and results for english senseval. *Computers and the Humanities*, 34:15–48, 2000.

[48] K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439, 1992.

[49] Nicholas Kushmerick. *Wrapper Induction for Information Extraction*. PhD thesis, University of Washington, 1997.

[50] Leah S. Larkey, M. Andrew Price Paul Ogilvie, and Brenden Tamilio. Acrophile: an automated acronym extractor and server. In *Proceedings of the ACM Fifth International Conference on Digital Libraries*, 2000.

[51] Chodorow M. Leacock C. *Combining local context and WordNet similarity for word sense identification.* MIT, 1998.

[52] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, 1986.

[53] Vladimir Levenshtein. Bynary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965.

[54] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.

[55] Dimitrios Mavroeidis, George Tsatsaronis, Michalis Vazirgiannis, Martin Theobald, and Gerhard Weikum. Word sense disambiguation for exploiting hierarchical thesauri in text classification. *Knowledge Discovery in Databases: PKDD 2005*, 3721/2005:181–192, 2005.

[56] Diana Maynard and Sophia Ananiadou. *Recent Advances in Computational Terminology*, chapter Term Extraction using a Similarity-based Approach, pages 271–278. John Benjamins, 2001.

[57] Luke K. McDowell and Michael Cafarella. Ontology-driven information extraction with ontosyphon. In *Proceedings of the 5th Internal Semantic Web Conference (ISWC'06)*, 2006.

[58] Matthew Michelson and Craig A. Knoblock. Unsupervised information extraction from unstructured, ungrammatical data sources on the world wide web. *International Journal on Document Analysis and Recognition*, 10(3):211–226, 2007.

[59] Rada Mihalcea. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of the HLT/EMNLP*, 2005.

[60] Rada Mihalcea and Dan Moldovan. A method for word sense disambiguation of unrestricted text. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, 1999.

[61] Rada Mihalcea, Paul Tarau, and Elizabeth Figa. Pagerank on semantic networks, with application to word sense disambiguation. In *Proceedings of the 20th international conference on Computational Linguistics*, 2004.

[62] George A. Miller, Richard T.Beckwith, Christiane D. Fellbaum, Derek Gross, and Katherine J.Miller. Wordnet: An on-line lexical database4. *International Journal of Lexicography*, 3(4):235–244, 1990.

[63] Steven N. Minton, Claude Nanjo, Craig A. Knoblock, Martin Michalowski, and Matthew Michelson. A heterogeneous field matching method for record linkage. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, 2005.

[64] Alvaro Monge and Charles Elkan. The field matching problem: Algorithms and applications. In *Proceedings of The Second International Conference on Knowledge Discovery and Data Mining, (KDD)*, 1996.

[65] T.F. Smith M.S. Waterman. Some biological sequence metrics. *Advances in Math*, 29(4):367–387, 1976.

[66] David Nadeau and Peter D. Turney. A supervised learning approach to acronym identification. In *Proceedings of the Eighteenth Canadian Conference on Artificial Intelligence*, 2005.

[67] Roberto Navigli and Mirella Lapata. Graph connectivity measures for unsupervised word sense disambiguation. In *Proceedings of ICJAI*, 2007.

[68] Saul Needleman and Christian Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[69] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:444–453, 1970.

[70] Youngja Park and Roy J. Byrd. Hybrid text mining for finding abbreviations and their definitions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2001.

[71] Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. Using semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics, Mexico City.*, 2002.

[72] Ted Pedersen, Serguei V.S. Pakhomov, Siddharth Patwardhan, and Christopher G. Chute. Measures of semantic similarity and relatedness in the biomedical domain. *Journal of Biomedical Informatics*, 40(3):288–299, 2007.

[73] Jakub Piskorski and Marcin Sydow. Usability of string distance metrics for name matching tasks in polish. In *Proceedings of the 3rd Language and Technology Conference, Poznan*, 2007.

[74] B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, A. Kirilov, and M. Goranov. Towards semantic web information extraction. In *Proceedings ofthe Human Language Technologies Workshop at 2nd International Semantic Web Conference*, 2003.

[75] James Pustejovsky, José Castaño, Brent Cochran, Maciej Kotecki, Michael Morrell, and Anna Rumshisky. Extraction and disambiguation of acronym-meaning pairs in medline. 2001.

[76] James Pustejovsky, José Castaño, Brent Cochran, Maciej Koteckib, and Michael Morrella. Automatic extraction of acronym-meaning pairs from medline databases. *Medinfo*, 10 (Pt 1):317–375, 2001.

[77] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *IEEE Transations on Systems,Man, and Cybernetics*, 19(1):17–30, 1989.

[78] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on artificial intelligence.*, 1995.

[79] Uwe Reyle and Jasmin Saric. Ontology driven information extraction. In *Proceedings of the 19th Twente Workshop on Language Technology*, 2001.

[80] Eric Sven Ristad and Peter N. Yianilos. Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5), 1998.

[81] David Sankoff and Joseph Kruskal. *TimeWarps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley, Reading, MA, 363-365, 1983.

[82] Ariel Schwartz and Marti Hearst. A simple algorithm for identifying abbreviation definitions in biomedical texts. In *Proceedings of the Pacific Symposium on Biocomputing (PSB). University of California, Berkeley.*, 2003.

[83] Ravi Sinha and Rada Mihalcea. Unsupervised graph-based word sense disambiguation using measures of word semantic similarity. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2007)*, 2007.

[84] Jiri Stetina, Sadao Kurohashi, and Makoto Nagao. General word sense disambiguation method based on al full sentential context. In *Proceeedings of the COLING/ACL '98 Workshop on usage of WordNet in NLP*, 1998.

[85] Michael Sussna. Word sense disambiguation for free-text indexing using a massive semantic network. In *Proceedings of the Second International Conference on Information and Knowledge Base Management, CIKM'93*, 1993.

[86] Kazem Taghva and Jeff Gilbreth. Recognizing acronyms and their definitions. *International Journal on Document Analysis and Recognition*, 1(4):191–198, 1999.

[87] J. R. Ullmann. A binary n-gram technique for automatic correction of substitution deletion, insertion and reversal errors in words. *The Computer Journal*, 20(2):141–147, 1977.

[88] Alexandros G. Valarakos, Georgios Sigletos, Vangelis Karkaletsis, Georgios Paliouras, and George A. Vouros. A methodology for enriching a multi-lingual domain ontology using machine learning. In *A. Valarakos, G. Sigletos, V. Karkaletsis, G. Paliouras, G.A. Vouros, "A Methodology for Enriching a Multi-Lingual Domain Ontology using Machine Learning", Proceedings of the Workshop "Text processing for Modern Greek: from symbolic to statistical approaches", 6th International Conference of Greek Linguistics*, 2003.

[89] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[90] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.

[91] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83, 1945.

[92] William Winkler and Y. Thibaudeau. An application fo the fellegi-sunter model of record linkage to the 1990 us decenial census. Technical report, Bureau of the Census, Washington, D.C., 1991.

[93] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *Proceedings of the 32nd annual meeting of the Association for Computational Linguistics. Las Cruces,*, 1994.

[94] Stuart Yeates. Automatic extraction of acronyms from text. In *Proceedings of the New Zealand Computer Science Research Students Conference University of Waikato*, 1999.

[95] Stuart Yeates, David Bainbridge, and Ian Witten. Using compression to identify acronyms in text. In *Data Compression Conference*, 2000.

[96] Burcu Yildiz and Silvia Miksch. Motivating ontology-driven information extraction. In *Proceedings of the International Conference on Semantic Web & Digital Libraries*, 2007.