Detection and integration of affective feedback into distributed interactive systems

Ovidiu Mircea Şerban



Institut National des Sciences Appliquées de Rouen

Laboratoire d'Informatique de Traitement de l'Information et des Systèmes

Universitatea "Babeş-Bolyai"

Facultatea de Matematică şi Informatică, Departamentul de Informatică

P H D T H E S I S

Speciality : Computer Science

Defended by

Ovidiu Şerban

to obtain the title of

PhD of Science of INSA de ROUEN

and "Babeş-Bolyai" University

Detection and Integration of Affective Feedback into Distributed Interactive Systems

The September 13, 2013

Jury :

Reviewers:				
Jean-Yves Antoine	-	*Professor*	-	"François-Rabelais" University
Dan Cristea	-	*Professor*	-	"Alexandru Ioan Cuza" University
Crina Groşan	-	*Associate Professor*	-	"Babeş-Bolyai" University
Examiner:				
Catherine Pelachaud	-	*Professor*	-	TELECOM ParisTech
PhD Directors:				
Jean-Pierre Pécuchet	-	*Professor*	-	INSA de Rouen
Horia F. Pop	-	*Professor*	-	"Babeş-Bolyai" University
PhD Supervisors:				
Alexandre Pauchet	-	*Associate Professor*	-	INSA de Rouen
Alexandrina Rogozan	-	*Associate Professor*	-	INSA de Rouen

"Man is a robot with defects."

– Emil Cioran

To Alina, and my family:
Mircea, Sorina and Sergiu
Thank you for all your support ...

Acknowledgements

There is a moment in every person's life when it has to compile a long list of names and to thank each one individually. Some of these people inspired me with their dedication to the academic life and most important their research work. Several people helped me along the way and made all the possible efforts that I can write this list. I am grateful for all the research opportunities I was involved and to all I worked with.

First of all, I would like to thank my two Ph.D. directors, prof. Jean-Pierre Pécuchet and prof. Horia F. Pop, for their guidance through these three years. I could not manage to do all this work without the two Universities that hosted me during my thesis: INSA de Rouen in France and "Babeş-Bolyai" University in Romania.

Second, I would like present my gratitude to the jury that accepted to review my thesis: prof. Jean-Yves Antoine ("François-Rabelais" University), prof. Dan Cristea ("Alexandru Ioan Cuza" University), Crina Groşan ("Babeş-Bolyai" University) and Catherine Pelachaud, TELECOM ParisTech. This list includes my supervision committee from "Babeş-Bolyai" University: prof. Gabriela Czibula, Mihai Oltean and Crina Groşan, and from INSA de Rouen: Alexandre Pauchet and Alexandrina Rogozan.

I would like to address my full gratitude to Mihai Oltean and Alexandre Pauchet for their major influence during my research career. Dr. Oltean is the one who though to conduct research during all the projects and activities we conducted together, while dr. Pauchet helped me during all the major critical points of my Ph.D. research. Moreover, I would like to thank Marc Schröder for giving me plenty of interesting ideas during our discussion in Glasgow, at the International Conference of Social Intelligence and Social Signal Processing, Ginevra Castellano for hosting and helping me for my work on multimodal fusion, at the Birmingham University and Laura Dioşan for recommending me this Ph.D.

During the French ACAMODIA project, funded by the CNRS (PEPS INS2I-INSHS 2011), I had the pleasure to work with a very friendly and active team: Mme Emilie Chanoni and her two students Anne Bersoult and Elise Lebertois. I really enjoyed our work together and the results we obtained.

Nevertheless, I could not end this thesis properly without the financial support given

by the French Eiffel Scholarship, for which I express my full gratitude, as well.

Last, I would like to thank my friends and family for their support. This list includes (I hope I did not forget anyone): Florian, Guillaume, Zacharie, Amnir, Yadu, Gabriel, Daniel, Nicoleta, Stefan and Bianca, which involved in Ph.D. theses as well (in one way or the other), understood and supported me during my work. I want to thank as well William Boisseleau, that helped me with the development of my platform during his internship. My final thank will go to Alina and my family for their full support and understanding.

With all my respect,
Ovidiu Şerban
Rouen, 1st of July, 2013

Summary

Human-Computer Interaction migrates from the classic perspective to a more natural environment, where humans are able to use natural language to exchange knowledge with a computer. In order to fully "understand" the human's intentions, the computer should be able to detect emotions and reply accordingly. This thesis focuses on several issues regarding the human affects, from various detection techniques to their integration into a Distributed Interactive System.

Emotions are a fuzzy concept and their perception across human individuals may vary as well. Therefore, this makes the detection problem very difficult for a computer. From the affect detection perspective, we proposed three different approaches: an emotion detection method based on Self Organizing Maps, a valence classifier based on multi-modal features and Support Vector Machines, and a technique to resolve conflicts into a well known affective dictionary (SentiWordNet). Moreover, from the system integration perspective, two issues are approached: a Wizard of Oz experiment in a children storytelling environment and an architecture for a Distributed Interactive System.

The first detection method is based on neural network model, the Self Organizing Maps, which is easy to train, but very versatile for fuzzy classification. This method works only with textual data and it uses also an Latent Semantic Analyser (LSA) feature extraction algorithm with large dictionaries as support vectors. The issue is approached as a Statistical Machine Learning problem and the validation is conducted on a well known corpus for semantic affect recognition: SemEval 2007, task 14. This experiment leads to a classification model that provides a good balance between precision and recall, for the given corpus.

We continue on the same Machine Learning perspective, by conducting a multi-modal classification study on a Youtube corpus. The smile, as a gesture feature, is fused with several other features extracted from textual data. We study the influence of smile across different configurations, with a two level linear Support Vector Machine. This offers the possibility to study in details the classification process and therefore, we obtain the best results for the proposed corpus.

In the field of Emotion Detection the focus is mainly on two aspects: finding the

best detection algorithms and building better affective dictionaries. Whereas the first problem is tackled by the algorithms previously presented, we also focus on the second issue as well. We are decreasing the number of inconsistencies of an existing linguistic resource, the SentiWordNet dictionary, by introducing context. This is modelled as a context graph (*contextonyms*), built using a subtitle database. By applying our technique, we managed to obtain a low conflict rate, while the size of the dictionary is preserved. Our final goal is to obtain a large affective dictionary that can be used for emotion classification tasks. Decreasing the number of inconsistencies in this dictionary would directly improve the precision of the method using it.

The contextonyms are cliques in a graph of word co-occurrences. Therefore, these represent a strong semantic relation between the terms, similar to synonymic relation. The clique extraction algorithm used for this purpose was designed for building the contextonym graph, since none of the existing algorithms could handle large and dynamic graph structures. Our algorithm, the Dynamic Distributable Maximal Clique Exploration Algorithm (DDMCE), was successfully validated on various random generated databases.

From the system integration perspective, the problem of Child-Machine interaction is tackled through a storytelling environment. From the psychological perspective, this experiment is a validation of the interactive engagement between a child and a virtual character. The engineering aspects of this experiment lead to the development of a new Wizard of Oz platform (OAK), that allows online annotation of the data. Moreover, this environment helps on designing and building new reactive dialogue models, which can be integrated into our future system.

The second aspect of system integration is tackled by building a new architecture for a Distributed Interactive System. This is constructed around the idea of component based design, where the structure of the component is simple enough to allow the integration of any existing algorithm. The proposed platform currently offers several components for knowledge extraction, reactive dialogue management and affective feedback detection, among other classic components (i.e. Automatic Speech Recognition, Text to Speech). Moreover, all the algorithms previously presented can be integrated into this platform as different components.

Keywords: *Affective Feedback, Human-Computer Interaction, Emotion Detection, Contextualised Dictionaries, Storytelling Environment, Distributed Interactive Systems*

Résumé

L'Interaction Humain-Machine a évolué d'une perspective classique vers un environnement plus naturel, dans lequel les humains peuvent utiliser la langue naturel pour échanger des connaissances avec un ordinateur. Pour bien "comprendre" les intentions de l'humain, l'ordinateur doit être capable de détecter les émotions et de répondre en conséquence. Cette thèse porte sur plusieurs aspects de la détection des émotion humaines, en partant de différentes techniques de détection jusqu'à leur intégration dans un Système Interactif Distribué.

Les émotions sont un concept flou et leur perception par des individus humains peut aussi varier. Par conséquent, cela rend le problème de détection très difficile en informatique. Du point vue de la détection de l'affect, nous avons proposé trois approches différentes : une méthode à base de Cartes Auto-Organisatrices (*Self Organizing Maps - SOM*), un classifieur de la valence basé sur des caractéristiques multi-modales et un Séparateur à Vaste Marge (*Support Vector Machines - SVM*) et une technique pour résoudre les conflits dans un dictionnaire affectif (*SentiWordNet*). En outre, du point vue de l'intégration aux systèmes, deux questions sont abordées : une expérience de type Magicien d'Oz dans un environnement de narration d'histoires pour enfants et une architecture de Système Interactif Distribué.

La première méthode de détection est basée sur un modèle de réseau neuronal, les Cartes Auto-Organisatrices, qui est un modèle simple à entraîner, mais très efficace pour la classification floue. Cette méthode fonctionne uniquement avec des données textuelles et utilise également une Analyse Sémantique Latente (*Latent Semantic Analyser - LSA*) pour l'extraction des caractéristiques avec des grands dictionnaires. La question est abordée comme un problème d'apprentissage artificiel et la validation est effectuée sur un corpus pour la reconnaissance sémantique des émotions (SemEval 2007, Tâche 14). Cette expérience a conduit à un modèle de classification qui offre un bon équilibre entre précision et rappel pour le corpus donné.

En continuant de nous baser sur des méthodes d'apprentissage artificiel, nous avons réalisé une étude de classification multi-modale sur un corpus Youtube. Le sourire, comme caractéristique de geste, est fusionné avec plusieurs autres caractéristiques ex-

traites à partir de données textuelles. Nous étudions l'influence du sourire à travers différentes configurations, avec un SVM sur deux niveaux. Cela offre la possibilité d'étudier plus en détail le processus de classification et donc, d'obtenir de meilleurs résultats pour le corpus proposé.

Dans le domaine de la détection d'émotion, l'accent est mis principalement sur deux aspects : la recherche des meilleurs algorithmes de détection et la construction de bons dictionnaires affectifs. Ayant partiellement traité le premier problème dans une partie, nous nous concentrons également sur le deuxième problème. Nous diminuons le nombre d'incohérences dans une ressource linguistique existante, le dictionnaire SentiWordNet, en introduisant le contexte. Ce contexte est modélisé comme un graphe de contexte (*contextonymes*), construit en utilisant une base de données de sous-titres. En appliquant notre technique, nous avons réussi à obtenir un faible taux de conflits, alors même que la taille du dictionnaire est préservée. Notre objectif final est d'obtenir un grand dictionnaire affectif pouvent être utilisé pour des tâches de classification d'émotions. La diminution du nombre d'incohérences dans ce dictionnaire peut améliorer directement la précision des méthodes qui l'utilisent.

Les contextonymes sont des cliques dans un graphe de co-occurrences de mots. Par conséquent, ceux-ci représentent une relation sémantique forte entre termes, comme le fait également la synonymie. L'algorithme d'extraction de cliques utilisé a été conçu pour construire le graphe de contextonymes, puisqu'aucun des algorithmes existants ne permetait la manipulation de graphes de grande taille et dynamiques. Notre proposition, l'Algorithme Dynamique Distribuable pour l'Extraction de Cliques Maximales (DDMCE), a été validée avec succès sur diverses bases de données générées aléatoirement.

Du point de vue de l'intégration de systèmes, le problème de l'interaction enfant-machine est abordée à travers un environnement de narration d'histoires. Du point de vue psychologique, cette expérience est une validation de l'engagement interactif entre un enfant et un personnage virtuel. Les aspects techniques de cette expérience conduisent au développement d'une nouvelle plate-forme Magicien d'Oz (OAK), qui permet l'annotation en ligne des données. En outre, cet environnement contribue à la conception et la construction de nouveaux modèles de dialogues réactifs, qui peuvent être intégrés dans notre futur système.

Le deuxième aspect de l'intégration est abordé par la construction d'une nouvelle architecture pour un Système Interactif Distribué. Cette architecture est basée sur une conception à base de composants, où la structure d'une pièce est suffisamment simple pour permettre l'intégration d'un algorithme existant. La plateforme proposée offre actuellement plusieurs composants pour l'extraction de connaissances, une gestion réactive du dialogue et la détection du feedback affectif et parmi d'autres composants plus classiques, comme par exemple la reconnaissance automatique de la parole et la synthèse vocale. En outre, tous les algorithmes présentés précédemment peuvent être intégrés dans cette plate-forme comme différents composants.

Rezumat

Interacţiunea dintre om şi calculator porneşte de la modalităţile clasice şi migrează încet spre un mediu mult mai natural, unde omul are posibilitatea să utilizeze exprimarea orală pentru a relaţiona cu calculatorul. Pentru a „înţelege" pe deplin intenţiile umane, calculatorul ar trebui să fie capabil să detecteze emoţiile şi să reacţioneze corespunzător. Această lucrare se concentrează pe câteva probleme legate de emoţiile umane, acoperind aspecte legate de la detecţie până la integrarea tuturor acestor elemente într-un Sistem Interactiv Distribuit.

Emoţiile umane descriu un concept vag care variază de la un individ la altul. Din perspectiva calculatorului, acest lucru face ca problema detecţiei lor să fie mult mai dificilă. Pentru a rezolva problema detecţiei emoţiilor, propunem trei abordări diferite: o metodă de detecţie bazată pe Reţele cu Auto-Organizare (*Self Organizing Maps - SOM*), un clasificator de valenţă bazat pe caracteristici multi-modale antrenate cu o Maşina de Vectori Suport (*Support Vector Machines - SVM*) şi o tehnică de rezolvare automată a conflictelor semantice dintr-un cunoscut dicţionar afectiv (*SentiWordNet*). Totodată, din perspectiva tehnică a lucrării, propunem soluţii la două probleme legate de interacţiunea dintre om şi calculator: un experiment realizat într-un mediu de tip „Magicianul din Oz" (*Wizard of Oz*) pentru observarea reacţiilor unor copii la ascultarea unei poveşti şi o arhitectură pentru realizarea unui Sistem Interactiv Distribuit.

Prima metodă de detecţie propusă este bazată pe un model de reţele neuronale: Reţele cu Auto-Organizare. Aceastea sunt uşor de antrenat, dar foarte utile în cazurile de clasificare cu anotări vagi sau imprecise. Metoda funcţionează doar pentru date de tip text şi utilizează un Analizor Semantic Latent (*Latent Semantic Analyser - LSA*) pentru extragerea caracteristicilor bazate pe vectori suport de talie mare. Problema este abordată din perspectiva unui algoritm de instruire automată, iar validarea este efectuată pe un corpus propus pentru recunoaşterea semantică a emoţiilor: SemEval 2007, problema 14. Experimentul a dus la realizarea unui model de clasificare care oferă un echilibru între precizie şi acurateţe.

În continuare, în acelaşi context de instruire automata, realizăm un studiu pe o serie de caracteristici multi-modale extrase folosind un corpus Youtube. Zâmbetul, consid-

erat în acest caz un parametru numeric discret, e fuzionat cu o serie de alte caracteristici extrase din text. În acest experiment, studiem influenţa zâmbetului combinat cu alte caracteristici, antrenate cu o Maşină de Vectori Suport pe două nivele. Această configuraţie ne oferă posibilitatea de a studia în detaliu procesul de clasificare şi, totodată, de a obţine cele mai bune rezultate pe corpusul propus.

Domeniul de detecţie automată a emoţiilor umane e concentrat pe două axe principale: găsirea de algoritmi de detecţie performanţi şi construirea de dicţionare afective. Interesul nostru este axat pe ambele aspecte, iar prima problemă este abordată de algoritmii descrişi anteriori. Pentru a scădea numărul de conflicte semantice a unei resurse linguistice existente (dicţionarul SentiWordNet) am introdus un nou concept bazat pe contextul unui cuvânt. Acesta presupune construirea unui graf de context, numit contextonym, utilizând un corpus de subtitrări. Aplicând această tehnică am reuşit să scădem numărul de conflicte semantice, păstrând neschimbată talia dicţionarului. Scopul nostru final este să construim un dicţionar afectiv de mari dimensiuni, care poate fi utilizat în rezolvarea problemelor de clasificare automată. Scăderea numărului de inconsistenţe în acest dicţionar va creşte în mod direct precizia metodelor care îl folosesc.

Contextonymele sunt clicuri într-un graf bazat pe coocurenţe de cuvinte, acestea având o relaţie puternică între ele, similară sinonimiei. Algoritmul folosit pentru extragerea clicurilor a fost conceput pentru a construi graful de contextonyme, deoarece nici un algoritm existent nu putea face faţă volumului mare de date şi caracterului dinamic al acestora. Propunerea noastră, Algoritmul Dinamic şi Distributabil pentru Explorarea Clicurilor Maximale (*Dynamic Distributable Maximal Clique Exploration Algorithm - DDMCE)*, a fost validat cu succes pe mai multe baze de date conţinând grafe generate aleator.

Din perspectiva tehnică a tezei, problema interacţiunii dintre copil şi calculator este abordată prin prisma unui mediu de povestire digital. Din punct de vedere psihologic acest experiment constituie o validare a angajamentului în interacţiune dintre copil şi un personaj virtual. Aspectele tehnice ale acestui experiment au condus la dezvoltarea unei noi platforme (OAK), bazată pe o metodologie de tip „Magicianul din Oz", care permite totodată anotarea în timp real a datelor. Acest mediu ajută la prototipizarea şi construcţia unor noi modele reactive de dialog, care pot fi integrate în viitorul nostru sistem.

Al doilea aspect tehnic al acestei teze presupune construirea unei noi arhitecturi pentru un Sistem Interactiv Distribuit. Acesta este centrată în jurul ideii de modelare bazată pe componente, cu o structură suficient de simplă pentru a permite integrarea unor algoritmi existenţi. Platforma propusă oferă câteva componente pentru extragerea conceptelor, managementul reactiv al dialogului sau detectarea feedbackului afectiv, în plus faţă de componente clasice, oferite de alte sisteme, cum ar fi: Recunoaştere Vocală sau Sinteză Vocală. Toţi algoritmii prezentaţi anterior pot fi integraţi în acest sistem sub formă de componente.

Contents

1 Introduction **13**

1.1 Human-Computer Interaction 13
1.2 Original aspects of this thesis 15
1.3 Structure of the thesis 16
1.4 Publication list 17

I Detection of User's Affective Feedback **21**

2 Emotion detection in dialogues **23**

2.1 Context 24
2.2 Related work 25
2.2.1 Psychological Perspective 25
2.2.2 Engineering Perspective 25
2.2.3 Affect Detection Methods 26
2.3 Preliminaries 28
2.4 Emotion detection in short texts 28
2.5 The Emotion Classification Model 30
2.5.1 The Self-Organizing Map Algorithm 31
2.5.2 Preprocessing Step 33
2.5.3 Feature Extraction 33
2.5.4 Self-Organizing Map Model 34
2.5.5 Results 35
2.6 The dialogue characteristics 37
2.6.1 Youtube Opinion Corpus 38
2.7 Multi-modal Affect Detection 40
2.7.1 Classification Model 41
2.7.2 SVM Classifier 41
2.7.3 Training and testing sets 42

2.7.4 Feature extraction . . . 43
2.7.5 Results . . . 45
2.8 Discussion . . . 47

3 Affective linguistic resources for emotion detection 51
3.1 Context . . . 52
3.2 Related work . . . 53
3.2.1 Affective Dictionaries . . . 53
3.2.2 Maximal Clique Exploration . . . 54
3.3 DDMCE Algorithm . . . 59
3.3.1 Preliminaries . . . 59
3.3.2 Algorithm . . . 61
3.3.3 Implementation . . . 65
3.3.4 Experiments . . . 68
3.4 A new linguistic resource: affective contextonyms . . . 71
3.4.1 SentiWordNet . . . 71
3.4.2 Modelling context with contextonyms . . . 74
3.4.3 A new linguistic resource: an affective contextonym model for dialogues . . . 76
3.4.4 Validation . . . 79
3.5 Discussion . . . 81

II Affective Interactive Systems 83

4 Affective interaction 85
4.1 Context . . . 86
4.1.1 Wizard of Oz experiments and avatars . . . 86
4.1.2 Corpus collection . . . 87
4.1.3 ACAMODIA Project . . . 87
4.2 Related Work . . . 87
4.3 Scenario . . . 89
4.4 OAK . . . 90
4.4.1 Scenario formalisation . . . 91
4.4.2 Architecture . . . 92
4.5 Project results . . . 93
4.6 Discussion . . . 95

5 AgentSlang: A new platform for rich interactive systems 97
5.1 Context . . . 98
5.2 Related Work . . . 99
5.3 A new platform for Distributed Interactive Systems: MyBlock . . . 104
5.3.1 The principles of a Distributed Interactive System . . . 104
5.3.2 Description of the platform . . . 105

5.3.3 Distributed aspects of MyBlock . . . 108
5.3.4 Data oriented design . . . 110
5.3.5 Components, Services and Schedulers . . . 112
5.3.6 Example . . . 114
5.3.7 Performance . . . 116
5.4 Syn!bad . . . 118
5.4.1 Context . . . 119
5.4.2 Example . . . 119
5.4.3 Implementation . . . 120
5.5 AgentSlang . . . 123
5.5.1 System components . . . 123
5.5.2 Input and Output components . . . 124
5.5.3 Natural Language Processing and Dialogue Models/Components 127
5.5.4 Affective Feedback . . . 129
5.6 Discussion . . . 131

III Conclusion and Future Work 135

IV Appendix 141

A Syn!bad Syntax 143

B DDMCE: Running times for Static Graphs 145

C DDMCE: Running times for Dynamic Graphs 147

Bibliography 151

List of Figures

1.1 The thesis structure and the interaction between different parts 16

2.1 Emotional label distribution over the training corpus. 30

2.2 Valence distribution over the training corpus. 30

2.3 The training process for a SOM model. The σ_t radius describes the training neighbourhood for the current neuron $W_{i,j}$ 31

2.4 The dominant emotion visualisation, where darker zones represent stronger emotion. Colour legend: Anger, Disgust, Fear, Joy, Sadness and Surprise 35

2.5 The top dominant emotions visualisation, where larger areas of the same color represent more intense emotions. Colour legend: Anger, Disgust, Fear, Joy, Sadness, Surprise and white represents No Emotion 35

2.6 A multi-modal representation of the text and gesture interaction, with a segmented annotation. The segments s_i are chosen by the annotator based on their function, directly on the video sequence, without strictly overlapping the gesture or text track 41

2.7 The SVM hyperplane separation using a linear kernel and margin equations 43

2.8 The continuous smile function along with three different thresholds: 40%, 50% and 60% according to most common smile levels in the corpus. The signal has been already smoothed with an average windowed filter. The blue circles represent the intersection point between the signal and the threshold line. 45

3.1 Graph to tree transformation, done with DDMCE, using the Tomita et al.'s pivot selection [196]. The cliques found are presented in the Q set and the pivot at each step is presented in the highlighted boxes. 61

3.2 - 65

3.3 The distributed processing pipeline.The master process generates the message queue $\{M_1...M_k\}$. P_i represents the i-th processor which takes a message and generates the solution sub-tree attached to the corresponding M_j. In this setup we have k messages and n processors, with $k \gg n$. 66
3.4 Running time comparison for several graph size samples (n) and for different densities (ρ), given on an logarithmic scale representing time. . . . 71
3.5 A SWN [11] example, also containing the visualisation model. P states the positive degree, N the negative and O the objective one. 72
3.6 A fully annotated contextonym graph, representing the whole neighbourhood of the word '*heart*'. The labels are coloured according to their valence: blue for positive, red for negative, purple for mixed-value (conflictual valences) and light-grey for neutral. 77
3.7 The Opinion Lexicon (OL) [112] disagreement and *not found* rates in comparison to different strategies for SWN corrections 80
3.8 The MPQA Lexicon [207] disagreement and *not found* rates in comparison to different strategies for SWN corrections 80
3.9 The Harvard General Inquirer (HGI) Lexicon [186] disagreement and *not found* rates in comparison to different strategies for SWN corrections 81

4.1 The ACAMODIA scenario, formalised into 15 slides and some of parts of the story have some communication errors included 90
4.2 A simple example of a WOz scenario which can be used with OAK. The boxes represent states, whereas the rounded boxes are observations . . . 91
4.3 The pilot view of OAK 92
4.4 The two child views designed for OAK 93
4.5 A quantitative analysis of the results, for the selected group of children . 94
4.6 The disfluencies results, for the selected group of children 94

5.1 The general system view having a new layer that deals with affective feedback. 98
5.2 The MyBlock functional separation is done in three different levels, each having assigned a different set of responsibilities. Each level has a set of key words assigned, which describe the functions of that level. The levels marked with * are more linked to advanced concepts and would not make the object of a simple integration task. 106
5.3 The performance comparison for ZeroMQ and ActiveMQ, for message throughput representation. The representation is done on a logarithmic scale 110
5.4 The simplified diagram of MyBlock data objects 112
5.5 The performance comparison for SEMAINE and MyBlock, for message throughput representation. The representation is done on a logarithmic scale 118
5.6 A Syn!bad example, presented as an automaton 120

5.7 The Component Monitor displays the debug log, filtered by a selected level, and the component activation (in this case, the green component) 124
5.8 The Text Component has the ability to: to subscribe to multiple channels, display all the data received and send text input 125

List of Tables

2.1 Headlines from the training corpus, presented with dominant emotions . 29
2.2 Dominant class for coarse-grained representation 36
2.3 Results for each emotional class 36
2.4 The systems presented in the SemEval competition 37
2.5 Overall results 37
2.6 An example of transcription and smile presence from Youtube Database, video 7. The transcriptions and smiles are provided per segment basis. The bad words are censored with * 40
2.7 Results for the neutral vs non-neutral separation for the first level of classification and positive vs negative for the second level 46
2.8 Results for the positive vs non-positive separation for the first level of classification and neutral vs negative for the second level 46
2.9 Detection results for a three class HMM-based classifier, as extracted from Morency et al. [126] 47

3.1 Running times for random generated graphs. n is the number of nodes and ρ is the density. The values presented for DDMCE are the average of the #cliques and time(s) measures, whereas for CLIQUES algorithm the values are those presented in Tomita et al. article [196]. 69
3.2 Running times for static and dynamic graphs. n represents the number of nodes, ρ is the density and c the number of cores. All the times ($T_{dynamic}$ and T_{static}) presented are in seconds. The */dynamic* measure is defined by the equation 3.11. 70
3.3 Disagreement level, according to C. Potts [156], between SWN and several other corpora 73

5.1 A comparison of key features of existing State of Art Interactive Systems 103

5.2 A running time comparison between ActiveMQ and ZeroMQ platforms. The time presented is expressed in milliseconds and the message size represents the length of the sequence sent over the platform 109
5.3 A message throughput comparison between ActiveMQ and ZeroMQ platforms. The throughput presented is expressed in number of message per second and the message size represents the length of the sequence sent over the platform 109
5.4 The data standards for various conversational agents related functionalities and their W3C recommendation status 111
5.5 A running time comparison between SEMAINE and MyBlock. The time presented is expressed in milliseconds and the message size represents the length of the sequence sent over the platform 117
5.6 A message throughput comparison between SEMAINE and MyBlock. The throughput presented is expressed in number of message per second and the message size represents the length of the sequence sent over the platform 117
5.7 Syn!bad pattern examples, using the style definition features 122
5.8 The values assigned to each style according to the pattern definitions from Table 5.7 122
5.9 A summary of all existing AgentSlang components 131
5.10 A comparison of key features of AgentSlang and existing State of Art Interactive Systems 133

CHAPTER 1

Introduction

1.1 Human-Computer Interaction

The interaction paradigm assumes the continuous reciprocal influence between two individuals. From another perspective, it is a game where one acts whereas the other reacts. In computer engineering, the interaction takes place between a computer and an individual. Historically, this process evolved from a *one way* communication, where the computer was playing the role of executor, to a *bidirectional* communication, with the system as a communication partner. The communication protocol is changing, as well, from the classical master-slave perspective to a collaborative environment.

Traditionally, when the computer is just an executor, the input channels formalised as simple buttons are sufficient to ensure a good level of communication. Nowadays, the interaction becomes more intuitive. The inputs are a fusion of voice, gestures, postures or physical signals (such as acceleration, speed, orientation). The buttons still play a major role in this process, because of their accuracy, but the systems are slowly migrating towards naturalistic input.

From the computer perspective, the output changed from the classical text interfaces to graphical and then to more intuitive ones, such as dialogue oriented interfaces. The Ambient Intelligence (AmI) field models scenarios where a computer is able to control the environment in a natural way (i.e. switch on/off lights, execute daily tasks, communicate through spoken language) [52, 173]. Moreover, the computers may have a personified appearance (i.e. an Embodied Conversational Agent (ECA) [154, 40] or a Robot [64]) or even a personality [175]. More formally, we will use throughout this thesis the concept of *agent* to describe an intelligent entity, such as an interactive system.

Current approaches describe the human-computer interaction as a collaborative, dialogue-based, task [6]. Both interaction and dialogue task involve rich exchange of information between at least two peers. The richness of input refers to the possibility to use multi-modal exchange in the communication process: spoken language, gestures,

postures, vocalisations. The main difference between the two aspects is that, usually, the dialogue involves knowledge management, whereas the interaction can be strictly reactive. So far, softer reactive dialogue models have been proposed, without being sufficient to ensure a good communication level [136]. In our work, we refer to *dialogue* as a model where certain level of response planning and knowledge management is involved, and to *interaction* as a model that is mainly reactive.

Both dialogue and interaction models involve feedback detection and generation. Several levels of feedback can occur at any point through the information exchange process [37, 8]. These include: perception feedback (positive if the phrase can be transcribed, negative in case of failure), interpretation (positive if the phrase can be interpreted correctly according to the rules describing the system, negative in case of a misinterpretation) or execution feedback (positive if a satisfactory response is generated, negative otherwise). Out of these, a special category of feedback is represented by the human emotions. They do not act directly at a certain feedback level, previously described, but influence them all. For example, a negative emotion in the context of a perception failure can influence the response style. Instead of replying a simple phrase, such as: *"I am sorry, but I do not understand"*, the agent could also build a solution for the problem: *"I am sorry, but I do not understand. I would increase the volume of the microphone and let you try again."*. In this example, the system is able to detect frustration, as a negative emotion, and propose a solution instead of just giving the result error.

Affect Oriented Modelling

In this perspective, R. W. Picard is one of the first computer scientist working in Affective Computing, to offer a new point of view for engineers [151]. In order to make human-computer interfaces more interactive, the author proposed to integrate emotional models into existing approaches. The problem has been described not as a strict detection or simulation issue, but with very fuzzy boundaries. One of the challenges underlined by Picard is that such systems would balance the detection rate against the user's satisfaction.

According to Oxford Dictionary [142], the Online Edition, an emotion is a strong feeling deriving from one's circumstances, mood, or relationships with others. On the other hand, the opinions are the beliefs or views of a group or a majority of people. From a general perspective, an emotion is more complex and fuzzy than an opinion. Usually in opinion mining field, the literature refers to the valence (negative or positive) of a certain opinion [28], which is a simplified model of an affective intensity.

From the psychological perspective, P. Ekman proposed his original emotion model [56, 54] using only six basic emotions, considered as universal and recognizable all around the world: Anger, Disgust, Fear, Happiness, Sadness, Surprise. This work is the foundation of the Universality Theory [45], which states that all living beings express emotions in the same way. W. James is one of the two pioneers in the field of physiological perception of an emotion, foundation for most of the signal processing

techniques [92].

Even with some controversies in the area, several European inter-disciplinary organizations decided to launch ambitious projects such as HUMAINE Consortium [86], which aims at linking different research communities, all working on the idea of human centred research. The initial phase of the project finished with the release of the HUMAINE database [49], that contains a video corpus annotated with different schemes, among which a basic set of tools needed to analyse the data. The project continues as an excellence network, with a lot of researchers involved.

More recently, the SEMAINE consortium, as part of the HUMAINE Excellence Network, finished a project that focuses on the multi-modal detection aspect [175]. The Sensitive Artificial Listener (SAL), proposed by SEMAINE, is able to detect human emotions based on face gestures and several qualitative speech features. Besides this, the AVEC Challenges [179] propose a set of annotated corpora to solve the same issue.

1.2 Original aspects of this thesis

None of the previous presented works focuses on the semantic level. Humans, in everyday interaction, use natural language, among other modalities, to exchange information. The semantic level corresponds to the information transmitted, to **what** is being exchanged. The gestures, postures and vocal features are linked to the transmission style, or **how** things are being transmitted. We agree that the fusion of multi-modal features is a difficult task, but we also state that the semantic part of the communication has to play an important role into the detection process: the **how** and **what** have to be considered together.

Currently, the semantic context of affective words is very poorly exploited. When it is done, in most cases, manually annotated linguistic resources are proposed. We suggest an approach that deals with context in affective dictionaries which is generated automatically out of linguistic resources freely available over the web.

In order to make the machines "understand" human emotions [151], the algorithms that deal with affect detection and simulation need to be integrated into a system. Moreover, many of the problems regarding affect-oriented interaction systems are currently solved (partially or entirely). Therefore, the integration of all these components into a unified platform becomes critical. Our proposition, AgentSlang, is built around the idea of component integration and provides an architecture for this purpose. Many other important steps have to come, but even so, this platform remains one of the biggest contributions of this thesis.

Long Term Goal

The intelligence is characterised by the ability to acquire and apply knowledge and skills [142]. Building an intelligent agent, described by these abilities, is a very difficult task. Nevertheless, such a behaviour can be simulated by integrating feedback detection mechanisms, which would make the whole system more interactive [151].

The final goal of this work is to build a natural interactive environment, by using Embodied Conversational Agents or Robots. The approaches proposed are reactive, based on the semantic feature extraction and emotions detected in a multi-modal context. The usage of affects would increase the interactiveness of the system, while being able to provide real-time feedback for a dialogue model.

1.3 Structure of the thesis

This thesis investigates two main directions: algorithms used to detect user's emotional feedback or to build affective linguistic resources, and systems constructed around the interaction paradigm. Figure 1.1 presents a detailed structure of this thesis, with links between several sub-projects.

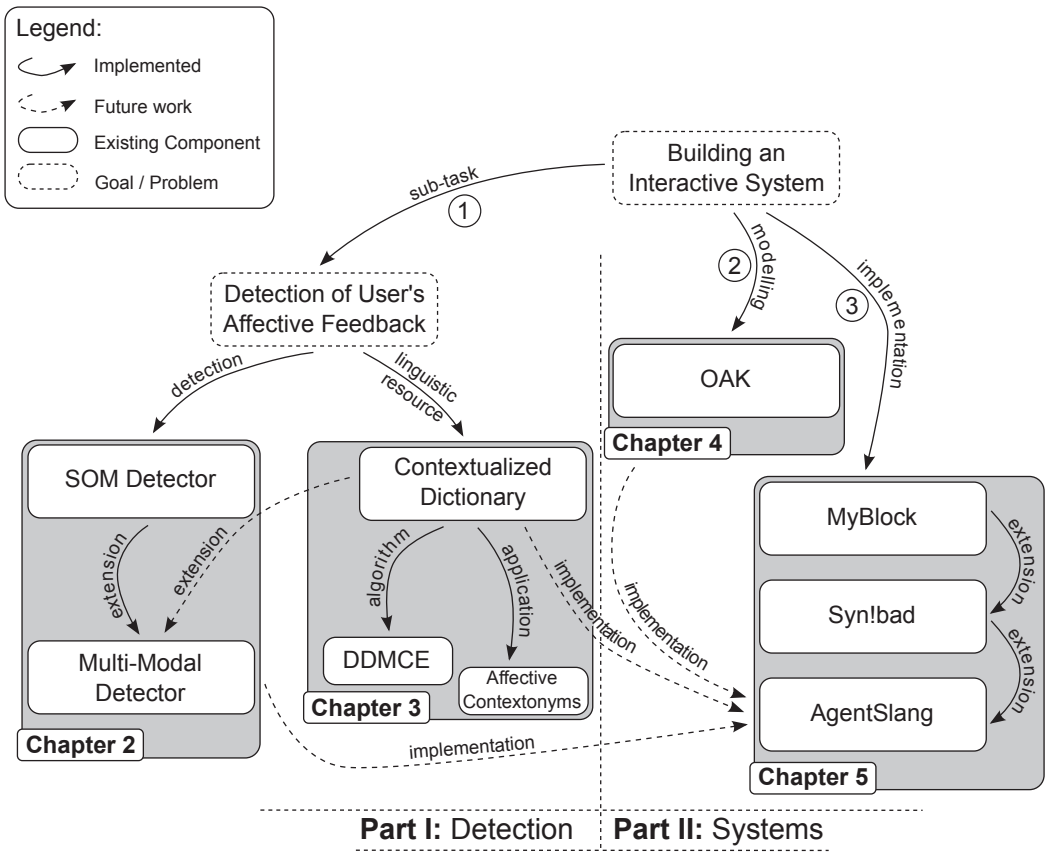


Figure 1.1: The thesis structure and the interaction between different parts

As stated before, the goal of this thesis is to build a system that deals with natural interaction. Several aspects of this problem need to be discussed:

(1) The problem of natural interaction is linked to emotions and affect detection. This is one of the major parts of this thesis.

(2) Due to the lack of rich interaction data, especially for children, an experiment dealing with corpus collection was performed.

(3) The design of an interactive system is proposed, which integrates several existing algorithms.

The problem of Affective Feedback Detection is tackled first by a Self Organizing Maps (SOM) Algorithm **(c10)**. This approach led to a second algorithm, which uses multi-modal features and a Support Vector Machine (SVM) to detect emotions **(c4)**. These models are described in **Chapter 2**.

While developing the SOM algorithm, we observed that the existing linguistic resources are not accurate enough to be used in affective detection tasks. Therefore, we developed a new methodology to create a context based affective dictionary **(c3)(c7)(p1)**. First, a new clique exploration algorithm was developed **(j1)(c8)**, which was applied afterwards on a subtitle corpora, annotated with SentiWordNet [11] valences. All these approaches are described in **Chapter 3**. In the future, this resource could be used as a dictionary for the multi-modal affect detector, described in **Chapter 2**.

On the System part, **Chapter 4** does a brief description of the protocol, formalised as a Wizard of Oz scenario, used to collect interaction data from a storytelling environment. Technical aspects of the experiment are described as well, by presenting the Online Annotation Toolkit (OAK) **(j2)(c2)(d1)**. Moreover, several psychological results are presented to support our hypothesis, that the interaction between a child and an avatar has similar characteristics with the interaction between a child and an adult in video conference mode **(c5)(c6)**.

Chapter 5 presents the architecture of the MyBlock, Syn!bad and AgentSlang projects **(c1)(c9)**. These are systems that allow easy modelling of component based design for building agents that deal with rich feedback data. Moreover, for the knowledge extraction part, the Syn!bad library is described.

In future, the interaction model computed from the data collected with the OAK platform would be compiled into a reactive model, which will be integrated into AgentSlang. The multi-modal emotion detection algorithm and the contextualized affect dictionary would be integrated as well into the same platform.

1.4 Publication list

Journal Papers

(j1) **O. Şerban**, A. Pauchet, A. Rogozan, and J-P. Pecuchet. DDMCE: a Dynamic Distributable Maximal Clique Algorithm, *Special Issue of International Journal of Social Network Mining*. 2013. (submitted)

(j2) **O. Şerban**, A. Bersoult, Z. Alès, E. Lebertois, É. Chanoni, F. Rioult and A. Pauchet. Modélisation de dialogues pour personnage virtuel narrateur, *Special issue of Revue d'intelligence artificielle*. 2013. (accepted)

Conferences

(c1) **O. Şerban** and A. Pauchet. AgentSlang: A Fast and Reliable Platform for Distributed Interactive Systems, *International Conference on Intelligent Computer Communication and Processing (ICCP)*. 2013.

(c2) **O. Şerban**, A. Pauchet, A. Bersoult, and E. Chanoni. Tell me a story: A comparative study of child interaction with virtual characters and adults, *Thirteenth International Conference on Intelligent Virtual Agents (IVA)*. 2013.

(c3) **O. Şerban**, A. Pauchet, A. Rogozan, and J-P. Pecuchet. Modelling context to solve conflicts in SentiWordNet, *The fifth biannual Humaine Association Conference on Affective Computing and Intelligent Interaction (ACII)*. 2013.

(c4) **O. Şerban**, G. Castellano, A. Pauchet, A. Rogozan, and J-P. Pecuchet. Fusion of Smile, Valence and NGram features for automatic affect detection, *The fifth biannual Humaine Association Conference on Affective Computing and Intelligent Interaction (ACII)*. 2013.

(c5) A. Pauchet, F. Rioult, E. Chanoni, Z. Ales and **O. Şerban**. Advances on Dialogue Modelling Interactive Narration Requires Prominent Interaction and Emotion, In Joaquim Filipe and Ana Fred, editors, *Proceedings of the 5th International Conference on Agents and Artificial Intelligence*, volume 1, pages 527–530, SciTePress, 2013.

(c6) A. Pauchet, F. Rioult, E. Chanoni, Z. Ales and **O. Şerban**. Modélisation de dialogues narratifs pour la conception d'un ACA narrateur, *Proceedings of the WACAI 2012 - Workshop Affect, Compagnon Artificiel, Interaction*, 8 pages, 2012.

(c7) **O. Şerban**, A. Pauchet, A. Rogozan and J-P. Pecuchet. Semantic Propagation on Contextonyms using SentiWordNet, *Proceedings of the WACAI 2012 - Workshop Affect, Compagnon Artificiel, Interaction*, 7 pages, 2012.

(c8) **O. Şerban**, A. Pauchet, A. Rogozan and J-P. Pecuchet. DDMCE : recherche de cliques maximales dans des graphes dynamiques de grande taille, *Proceedings of the 3ième Journée thématique : Fouille de grands graphes*, 5 pages, 2012.

(c9) Z. Ales, G. Dubuisson Duplessis, **O. Şerban** and A. Pauchet. A Methodology to Design Human-Like Embodied Conversational Agents, *Proceedings of the 1st International Workshop on Human-Agent Interaction Design and Models*, pages 34-49, 2012.

(c10) **O. Şerban**, A. Pauchet, and H.F. Pop. Recognizing emotions in short text. In Joaquim Filipe and Ana Fred, editors, *Proceedings of the 4th International Conference on Agents and Artificial Intelligence*, volume 1, pages 477–480. SciTePress, 2012.

Demo

(d1) **O. Şerban** and A. Pauchet. OAK: The Online Annotation Kit, *Proceedings of the WACAI 2012 - Workshop Affect, Compagnon Artificiel, Interaction*, 2 pages, 2012.

Poster

(p1) **O. Şerban**, Adding affective states to contextonyms, *International Workshop on Voice and Speech Processing in Social Interactions*, Glasgow, UK, 2011

Part I

Detection of User's Affective Feedback

CHAPTER 2

Emotion detection in dialogues

"Any emotion, if it is sincere, is involuntary."

– Mark Twain

Contents

2.1 Context 24
2.2 Related work 25
2.2.1 Psychological Perspective 25
2.2.2 Engineering Perspective 25
2.2.3 Affect Detection Methods 26
2.3 Preliminaries 28
2.4 Emotion detection in short texts 28
2.5 The Emotion Classification Model 30
2.5.1 The Self-Organizing Map Algorithm 31
2.5.2 Preprocessing Step 33
2.5.3 Feature Extraction 33
2.5.4 Self-Organizing Map Model 34
2.5.5 Results 35
2.6 The dialogue characteristics 37
2.6.1 Youtube Opinion Corpus 38
2.7 Multi-modal Affect Detection 40
2.7.1 Classification Model 41
2.7.2 SVM Classifier 41
2.7.3 Training and testing sets 42
2.7.4 Feature extraction 43
2.7.5 Results 45
2.8 Discussion 47

2.1 Context

Interactive Systems become more and more popular, therefore rich interaction between humans and computers is considered a priority. In order to increase the interactivity of the existing systems, the computers "need to understand" human emotions [151].

Concerning textual interaction, emotion detection has been approached by several research groups, each of them with its own specificity. In the field of Sentiment Analysis and Emotion Detection based on text data, two main directions for research exist: one concentrating on building better annotations of linguistic resources, such as dictionaries or ontologies [188, 11], and the other on building better classifiers for valence, sentiment or emotion detection [198, 187, 28]. This chapter focuses on classification approaches to detect emotion or valence in dialogue data. Several surveys [28, 73, 134] study these classifiers and group them based on: classification labels (continuous or discrete), methods and features (text or multi-modal).

Most of these methods base their approach on large collections of annotated documents, which serve as a training set. In a more specific scenario, such as dialogue systems, the length of a phrase is shorter than in publicist style, even if this is linked sometimes to oral communication or to a very familiar style. The text recovered from blogs, online comments, forums, journals or books, is more likely to be rigid, and respecting a formal style of writing and quality. Chat and dialogue, on the other hand, are, most of the time, the opposite of this. They are both represented as a written text, because dialogue is in most of the cases transcribed, the style and norms are not the same.

The chat scenarios we envision are modelled as an interaction between two or multiple parties, by using any software application that allows this type of communication. It also allows the recovery of the exchanged information (i.e. chat logs). The chat uses, most likely, a very specialised internet language, created to shorten the exchanged information to the minimum, called internet slang (i.e. Internet short-hand, leet, netspeak or chatspeak). Most of the chat would have emoticons available, which are a set of text conventions which can be translated into a set of animated or non-animated faces, representing moods, feelings, states of mind or communication gestures (i.e. waiving).

The dialogue data consists in any transcribed text log, as the result of verbal interaction between one or multiple parties. These logs can be annotated with the identity of the speaker or this information can be recovered from the voice track. Usually, the transcription does not use the same internet slang nor emoticons as a chat system does. The recovery of these logs could be done online (when the information is available in real-time) or offline, structured as a corpora.

In the following section, we present the state of art on Affective Computing [151], starting with several psychological and engineering aspects of the problem. We continue with the engineer's perspective and a short survey of the most important methods used for affect detection. The following sections cover the emotion classification model we proposed, based on a Self-Organizing Maps algorithm and applied on the SemEval corpus. A multi-modal affect detection approach applied in a dialogue context is in-

troduced afterwards by presenting an SVM classifier, applied on a Youtube corpus. At last, we conclude this chapter with a brief discussion.

2.2 Related work

2.2.1 Psychological Perspective

Paul Ekman proposed an original emotion model [56, 54] using only six basic emotions, considered as universal and recognizable all around the world: Anger, Disgust, Fear, Happiness, Sadness, Surprise. However, he expanded this list in a revision of his initial work [59], by introducing 12 new emotions: Amusement, Awe, Contempt, Enjoyment, Embarrassment, Excitement, Guilt, Pride in Achievement, Relief, Satisfaction, Sensory pleasure, Shame. These emotional are not meant to be exclusive. In fact, in more complex models, these emotions may occur in various intensities and different combinations. Paul Ekman also developed in collaboration with W.V. Friesen [57] and Dr. E. Rosenberg [58] the Facial Action Coding System (FACS), an annotation system dedicated to areas where emotion detection is considered critical.

Even if the universality of these emotions has been documented by multiple studies [56, 54, 92], some psychologists, such as J. A. Russell [165], noticed minor contradictions. Russell observed social and cultural situations where emotions are altered to something different from the universal belief; he explained that certain communities do not have an equivalent for the English word "fear". In the defence of the universal theory, other authors [65], found relations between emotions and language, as speech is considered as one of the vital aspects of affect expression, but not the only one, and created models where the social relations and background of individuals are the most influential aspects of emotion detection and expression. In this context, Russell's observation is a problem of linguistic expressiveness, as some words could have only a meta-linguistic equivalent, such as a gesture or interjection.

More recently, Scherer et al. [171] defines the emotions as "dynamic episodes characterized by a high degree of coordination between several organismic subsystems in the interest of optimal adaptation to relevant events". Moreover, the emotions have features related to specific events, affect most of the body functions in a synchronised way, are subject to rapid change and have a strong impact on behaviour generation [172].

2.2.2 Engineering Perspective

William James is one of the two namesakes of the James-Lange theory of emotions [92], which he formulated independently of Carl Lange in the 1880s. They were pioneers in the field of physiological perception of an emotion, foundation for most of the signal processing techniques.

In this perspective, R. W. Picard is one of the first engineers working in Affective Computing, offering a new point of view to engineers and computer scientists [151]. In order to improve human-computer interfaces, she proposed to integrate emotion detection engines. She described the problem not as a strict detection issue, but one

with very fuzzy boundaries. One of the challenges underlined by Picard is that such systems should balance the detection rate against the user's satisfaction.

From the Computer Science perspective, emotions are encoded usually into a numerical representation, by defining the probability of a certain emotional state or opinion to occur. In some cases, a corpus is annotated with the presence/absence of an emotional state, without any reference to the occurring probability. Therefore, a computer should be able to detect the affects in both these cases.

2.2.3 Affect Detection Methods

Ochs et al. [134] presents a series of systems that can be used for emotion detection, which exploits acronyms, emoticons, most common spelling errors and internet slang. These approaches use dictionaries and statistical machine learning to represent their features. Unfortunately, none of these algorithms are publicly available and they do not deal with dialogue data.

Calvo et al. [28] presents a comparative study of the algorithms used in Affective Computing, which are grouped by modality: visual features, vocal or text. Each modality is processed independently. Gunes et al. [73] presents a survey for the multi-modal feature fusion, mainly from the visual and acoustic perspective.

As a synset database example, we mention WordNet Affect [188], an extension of the WordNet [123] data set. WordNet Affect is basically a 6 class emotional annotation (i.e. Ekman's basic annotation scheme) made on a synset[1] level; it contains nouns, adjectives, adverbs and some verbs for the English WordNet 2.0 version. ConceptNet [113] is another well-known ontology widely used for semantic disambiguation in classification tasks. This database contains assertions of commonsense knowledge encompassing the spatial, physical, social, temporal, and psychological aspects of everyday life. ConceptNet was generated automatically from the Open Mind Common Sense Project. Another database, used especially for opinion and valence classification, is SentiWordNet [11] which is the result of automatic annotation of all WordNet synsets according to their degrees of positivity, negativity, and neutrality.

Starting from WordNet Affect, Valitutti et al. [198] proposed a simple word presence method in order to detect emotions. Ma et al. [116] designed an emotion extractor from chat logs, based on the same simple word presence. SemEval 2007, task 14 [187] presented a corpus and some methods to evaluate it, most of them based on Latent Semantic Analyser (LSA) [53] and WordNet Affect presence [187].

Methods more related to signal processing were proposed by [9], [44], or [47] which introduces different solutions in the classical approach of the field, for feature extraction, selection and classifiers. Alm et al. [9] used a corpus of child stories and a Winnow Linear method to classify the data into 7 categories. Using the ISEAR [204] dataset, a very popular collection of psychological data around 1990, [44] used different classifiers such as Vector Space Model (VSM), Support Vector Machine (SVM) or Naive-Bayes (NB) method to distinguish between 5 categories of emotions.

[1] A WordNet synset is a synonym collection attached to a sense of a word

The research involving text content concerns the detection of any kind of emotion in written text or transcripts of oral communication. The first approaches were done by some psychologists and anthropologists that were able to annotate and evaluate the semantic value of certain words. Charles E. Osgood conducted experiments to see how certain emotions can be inducted through text [137]. In order to measure the affective state, he used multidimensional scaling (MDS [22]) to create a visualisation of the affective space, based on the similarities between subjects from different cultures [138].

Morency et al. [126] builds one of the first multi-modal corpora dealing with text, voice and visual features. The prediction is done at the video level, and the text features are extracted using an affective dictionary. The gestures considered for the task are smile and eye gazing, which is associated usually with energy rather than emotion.

The AVEC Challenges [179] are multi-modal affect detection competitions, which provide a unique corpus, a set of features already extracted and a baseline comparison algorithm. Unfortunately, the transcription language proposed for the latest corpus is not consistent with the existing text dictionaries. This makes the design of a semantic extraction algorithm very difficult.

One of the most influential articles in affective computing, written by Gunes and Schuller [73], argues that the affective human behaviour analysis should be done in a continuous input setting, since it simulates well the natural environment. The continuity setting refers to a model able to classify segment or frame based inputs. Moreover, a model based on valence (the degree of positivity/negativity), arousal (being awake or reactive to stimuli) and energy (the power expressed by the actor), should be preferred. By using this model, detecting valence properly increases the precision of any multi-modal affect detection system.

Unfortunately, in the past AVEC Challenges [179] the participants preferred to use the spontaneous voice and visual features, rather than using semantic characteristics. Our approach defends the importance of the semantic level in the task of affective information detection and highlights that some gestures, such as smile, boosts the detection rate. Moreover, we concentrate our approach on segment based detection, rather than predicting a single label for a whole video sequence. Our approach models a text and gestures fusion, by integrating low level features extracted from dialogue data with visual features, such as smile. The corpus used in our experiments is the one proposed by Morency et al. [126] in their experiments, because it models a dialogue situation and offers the opportunity to compare the results.

In the following sections, we present the two problems (i.e. chat and dialogue data) as being the same, because despite some specific pre-filtering steps, both data sources consist in short sentences, which can refer a dialogue context. We present our contributions on detection of emotional labels in short texts and a fusion approach, suitable for multi-modal dialogue systems.

2.3 Preliminaries

In order to preserve a certain uniform distinction between the technical terms used across this thesis, we introduce a set of definitions, as present in the online edition of the Oxford Dictionary[2]:

> **Word**: A single distinct meaningful element of speech or writing, used with others (or sometimes alone) to form a sentence and typically shown with a space on either side when written or printed.
>
> **Term**: Logic of a word or words that may be the subject or predicate of a proposition.
>
> **Occurrence**: The fact of something existing or being found in a place or under a particular set of conditions.

Accordingly, the item **word** is used either as a part of a phrase or by its more generic term, as an occurrence in a dictionary.

2.4 Emotion detection in short texts

In the area of dialogue systems and feedback detection, short text is represented mainly by the phrases exchanged between the dialogue partners. Finding a proper annotated corpus to train and test different statistical machine learning technique proves to be a very difficult task. Our first choice is a corpus of short sentences, consisting in news headlines, extracted from various websites and provided by Strapparava and Mihalcea [187].

Emotions Corpus

The chosen corpus for our experiment was proposed for SemEval 2007, task 14 [187], published at the conference with the same name. The data set contains headlines (newspaper titles) from major websites, such as New York Times, CNN, BBC or the search engine Google News. The corpus was chosen since its characteristics and structure suits our problem requirements and we could easily compare our results with other systems that participated to the SemEval task.

The corpus was manually annotated by 6 different persons. They were instructed to annotate the headlines with emotions according to the presence of affective words or group of words with emotional content. The annotation scheme used for this corpus is the basic six emotions set, presented by Ekman: Anger, Disgust, Fear, Joy (Happiness), Sadness, Surprise. In situations were the emotion was uncertain, they were instructed to follow their first feeling. The data is annotated with a 0 to 100 scale for each emotion.

A valence annotation was also carried out. Valence, as used also in psychology, means the intrinsic attractiveness (positive valence) or aversiveness (negative valence)

[2] `http://www.oxforddictionaries.com/`

of an event, object, or situation. In SemEval task, the valence is used to describe the intensity of the positive or negative emotion. The valence label ranged from -100 to 100.

The recommendation for the task competition was to classify the data in an unsupervised manner. The corpus was split into testing and training section. Participants were allowed to create their own resources, without any obligation to share them with other competitors. To correlate the data from the 6 annotators, the authors established an inter-annotator agreement which resulted with a Pearson correlation coefficients. It enables to adjust the results obtained by each system with the data labelled by the annotators.

Table 2.1 presents examples of headlines from the corpus, among with their significant emotions. The scale of the emotions was normalized by 100 (translating all the values into a -1.0 to 1.0 interval), and the significant emotions were chosen in the neighbourhood of the dominant emotion (all the values between 20% range).

A	**D**	**F**	**J**	**Sad.**	**Sur.**	Headline
-	-	-	0.15	0.25	-	Bad reasons to be good
-	-	-	-	-	0.36	Martian Life Could Have Evaded Detection by Viking Landers
-	-	0.68	-	-	-	Hurricane Paul nears Category 3 status
-	-	-	0.75	-	0.57	Three found alive from missing Russian ship - report
0.52	0.64	0.50	-	0.43	-	Police warn of child exploitation online

Anger=**A**, Disgust=**D**, Fear=**F**, Joy=**J**, Sadness=**Sad.**, Surprise=**Sur.**

Table 2.1: Headlines from the training corpus, presented with dominant emotions

The authors of the corpus proposed a double evaluation, for both valence and emotion annotated corpus, on a fine-grained scale and on coarse-grained scale. For the fine-grained scale, for values from 0 to 100 (-100 to 100, for valence), the system results are correlated using the Pearson coefficients computed in the inter-annotator agreement. The second proposition was a coarse-grained encoding, where every value from the 0 to 100 interval is mapped to either 0 or 1 (0 =[0,50) , 1=[50,100]). Considering the coarse-grained evaluation, a simple overlap was performed in order to compute the precision, recall and F-measure for each class.

Another important aspect of this corpus is the emotion distribution inside the data set. In the figures 2.1 and 2.2, some dominant classes can be observed, as the negative class for the valence. For emotions, Sadness, Joy and Fear are the dominant clusters, since the intensity should be as high as possible (lower intensities are influenced by annotation noise). These are easy to annotate by humans experts, as it can be observed in the inter-annotator agreement [187].

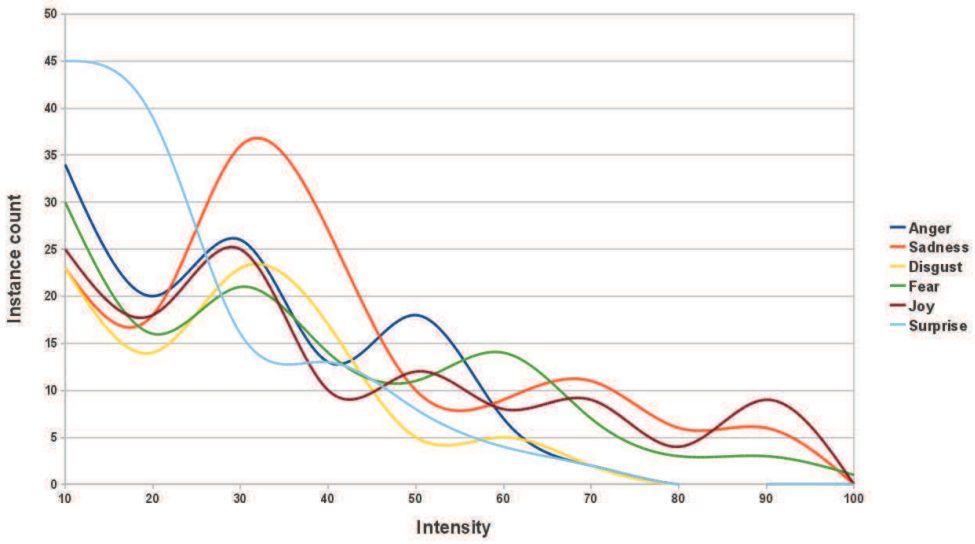


Figure 2.1: Emotional label distribution over the training corpus.

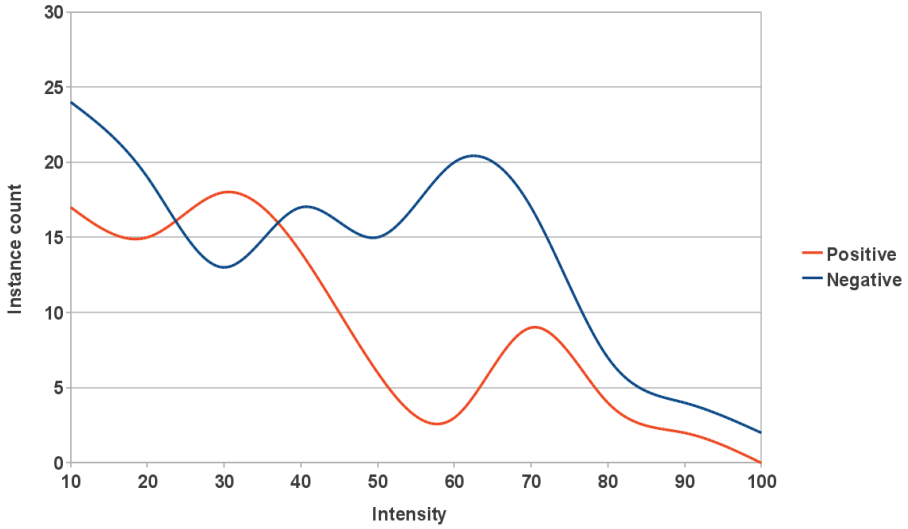


Figure 2.2: Valence distribution over the training corpus.

2.5 The Emotion Classification Model

The classifier we have chosen is a commonly used unsupervised method, the Self-Organizing Map (SOM) [105]. This method, proposed by the Finish professor Teuvo Kohonen, and sometimes called Kohonen maps or self-organizing feature map, is a particular type of neural network used for mapping large dimensional spaces into small dimensional ones. The SOM has been chosen because: 1) it usually offers good results in fuzzy data, 2) the training process is easier than with other Neural Networks and 3) the classification speed is sufficiently high for our problem. We start this section by introducing the SOM algorithm.

Our technique requires a multi-step process, each step assuring the output for the next phase. The first step, also called preprocessing, consists in filtering and cleaning the text information. The feature extraction and a projection follows, by using multiple

LSA strategies. In the third step, the SOM algorithm is applied and the trained model is used in the classification step. The two first steps are used both for the training and testing corpus, whereas the SOM algorithm is applied only during the training phase.

2.5.1 The Self-Organizing Map Algorithm

The SOM is a special type of neural networks used for unsupervised training. As for any machine learning algorithm, the data is split into training and testing. The data samples are defined as: $\mathcal{X} = \{x_k | x_k \in \mathbb{R}^n\}$, for the training set and $\mathcal{Y} = \{y_k | y_k \in \mathbb{R}^n\}$ for testing.

The method uses a grid configuration of neurons, where each one is connected to its nearby neighbours. The neurons are weights ($W_{i,j} \in \mathbb{R}^n$), initialised with random values, that need to be fitted by the training algorithm. The size of the network is defined as n_{som} for the width and m_{som} for the height.

The training process is iterative and the number of iterations T can be decided as the maximal point where the model begins to overfit the training data. Usually, this parameter is computed empirically over a series of training, while the overfitting error is computed through a cross validation. Figure 2.3 describes the training process of a SOM model.

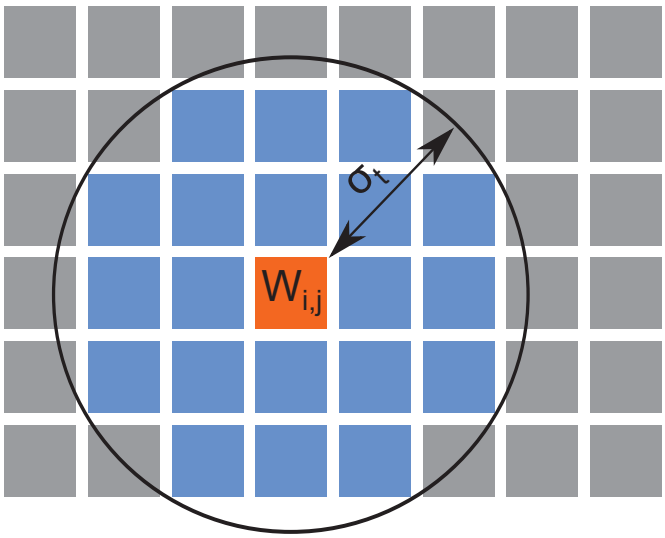


Figure 2.3: The training process for a SOM model. The σ_t radius describes the training neighbourhood for the current neuron $W_{i,j}$

The central piece of the training and classification algorithm is the Best Matching Unit (BMU) measure. The BMU is computed as an Euclidean distance over two given individuals, neurons or data sample. The equation for the BMU distance ($dist_{bmu}(a, a'); a, a' \in R^n$) is given by the following (equation 2.1):

$$dist_{bmu}(a, a') = \sqrt{\sum_{i=1}^{n} (a_i - a'_i)^2} \tag{2.1}$$

At each iteration (t), for every new training sample ($x_k \in \mathcal{X}$), the BMU (neuron

$W_{i,j}$) associated to x_k is defined as by the following:

$$BMU(x_k) = \min_{W_{i,j}} dist_{bmu}(x_k, W_{i,j}) \tag{2.2}$$

For a given neuron ($W_{i,j}$), the training radius (σ_t) is define as:

$$\sigma_t = \sigma_0 \times \exp(-\frac{t}{\lambda}) \tag{2.3}$$

$$\sigma_0 = \lambda = \frac{T}{\log(\frac{\max(n_{som}, m_{som})}{2})} \tag{2.4}$$

where: t is the current iteration of the training process, T is the total number of iterations and n_{som}, m_{som} are the width and height or the network.

This radius defines the σ_t neighbourhood, where the BMU training influence is spread. To complete the training process, the following equation is defined for all the neurons ω, contained in the neighbourhood of the BMU (the neuron $W_{i,j}$):

$$\omega_{t+1} = \omega_t + \Theta_t \times L_t \times (x_k - \omega_t) \tag{2.5}$$

where the Learning Rate, L_t, is:

$$L_t = L_0 \times \exp(-\frac{t}{T}) \tag{2.6}$$

and the BMU Influence Rate, Θ_t, is:

$$\Theta_t = \exp(-\frac{dist_{bmu}(\omega, W_{i,j})^2}{2 \times \sigma_t^2}) \tag{2.7}$$

The BMU has two roles in this algorithm: 1) as the best fitting neuron for a given sample and 2) it spreads the information already learned across nearby neighbourhood. This speeds up the training process and allows the creation of neuron clusters having the same properties. The Learning Rate and σ neighbourhood decay over time, which allows the learning process to concentrate over smaller parts of the network and to create better fittings for the data. Moreover, the BMU Influence Rate decays over the distance from the BMU, which allows the neighbouring neurons to learn at a higher rate, whereas distant ones still preserve the information they learned independently.

For the classification part, we used the same measure as during the training phase, which computes a distance from a proposed individual to all the elements in the SOM grid. The Best Matching Unit ($BMU(y_k)$) is selected, i.e. the element of the grid which is closest to the desired individual.

This training algorithm uses the numeric features extracted from the training and testing data. This process of extraction is described by the following sections.

2.5.2 Preprocessing Step

During the preprocessing step, we applied on each headline a collection of filters, in order to remove any useless information, such as special characters and punctuation, camel-case separators and stop word filtering. We considered as stop words, all prepositions, articles and other short words that do not carry any semantic value. The stop word collection used in our experiment is available at: `http://www.textfixer.com/resources/common-english-words.txt`. To reduce the space, we kept only the words that are considered to carry a strong semantic and emotional value, as WordNet Affect is suggesting [188].

This method offers a good balance between speed and accuracy of the results, compared to other methods such as Part of Speech Tagging (POS), which provides comparable results, but tends to be slower.

2.5.3 Feature Extraction

From the feature extraction perspective, we have chosen a Latent Semantic Analysis (LSA), applied with three different strategies. LSA is a well known strategy in Natural Language Processing field for measuring similarities between multiple documents and collections of terms. LSA assumes that words semantically close can be found close together in texts. Hence, all the occurrences of key terms are counted and introduced in a matrix (a row for each keyword, a column by document or paragraph). A Singular Value Decomposition (SVD) is applied on the resulted matrix in order to obtain the weighted similarities. In our experiments, the document collection is represented by the headlines corpus, where each headline is a separate document and the term set is chosen according to three different strategies.

The first LSA strategy we implemented concerns the algorithm applied onto the words of the WordNet Affect database [188]. This method is called by C. Strapparava and R. Mihalcea pseudo-LSA or meta-LSA [187]. The meta-LSA algorithm differs from the classic implementation by using clusters of words instead of single words in the LSA algorithm. The clusters are formed by the WordNet Affect synsets, available in the database. We expect the recall of the classifier to increase, but the precision to be low, which confirms the results of R.Mihalcea and C. Strapparava.

This strategy did not provide the expected results: the recall decreased since all of the presented words were carrying an emotional value and the non-emotional words were not represented. Our implementation confirms the results obtained by R.Mihalcea and C. Strapparava.

The second set of features was still extracted using the classic LSA, but applied onto the words of the training set. This strategy aims at refining the word collection in order to fully qualify all the input data. Although the genericness of this approach is not assured by the support word collection, this method offers a good starting point in document similarity experiments when the testing and training corpus are similar.

Our third proposition was to use the top 10 000 most frequent English words (except the stop words), extracted from approximately 1 000 000 documents existing in the

Project Gutenberg[3]. This corpus has been chosen as the largest free collection available and it offers a clear image over the English language. The words are used as key terms in the k-LSA strategy [53].

The features used are the document similarities obtained after applying the LSA algorithm. The SVD decomposition is applied on X, the initial occurrences matrix:

$$X = U * \Sigma * V^T \tag{2.8}$$

The k-LSA version eliminates the null values from the Σ diagonal matrix and k is the reduction index. The resulting matrix becomes X^k, which is a sub-matrix of size k of the initial matrix X. In practice, after applying the k-LSA algorithm, the X^k will be used as the X matrix in future computations.

After the feature extraction, the feature selection is performed in order to limit the feature space, which is done automatically with the k-LSA algorithm.

For the training part, the feature vectors are the columns from the V^T matrix, which represent the document similarities. For testing, a feature projection is done by translating the new occurrence matrix into the document space:

$$X' = \Sigma^{-1} * U^T * X \tag{2.9}$$

where X is the occurrence matrix computed on the testing corpus.

2.5.4 Self-Organizing Map Model

Many of the proposed implementations of the Self-Organizing Map use the feature model or a linear combination of the features for classification. Our implementation is very close to the classic one, but the feature space and classes were split into two distinct concepts and the classes are not used actively in the self-organizing algorithm; **data** and **label** vectors are separated in the Self-Organized Nodes and the learning process is done similarly for both of the vectors, with the same parameters.

During our experiments, a 40×40 grid size was used for the SOM configuration. The feature vectors were the document similarity vectors obtained from the feature extraction step, i.e. the columns of the V^T matrix obtained in the SVD decomposition from the LSA algorithm. As for the labels, we used the intensities available in the corpus as an independent vectorial space. The SOM technique was performed only for the projection of the original data into a bi-dimensional space and the actual classification was done by another step, described in the next section.

Because this method is mainly used in a visual manner to classify the results, we also built a visualisation module for easier evaluation of the method. Since a complex labelling technique was used, with 6 emotions represented by their probability of occurrence, a color was assigned to each emotion. The results can be interpreted in two ways: the representation of only one dominant emotion (figure 2.4) and the representation of

[3]Project Gutenberg is a large collection of e-books, processed and reviewed by the project community. All the documents are freely available at the website: `http://www.gutenberg.org/`

the most dominant emotions (figure 2.5).

Figure 2.4 represents the dominant emotion, after the learning process is finished. Darker zones represent more intense emotions. This visualisation shows that by using the SOM algorithm, the result converges to a structured model. This visualisation is inspired by the original SOM article [105].

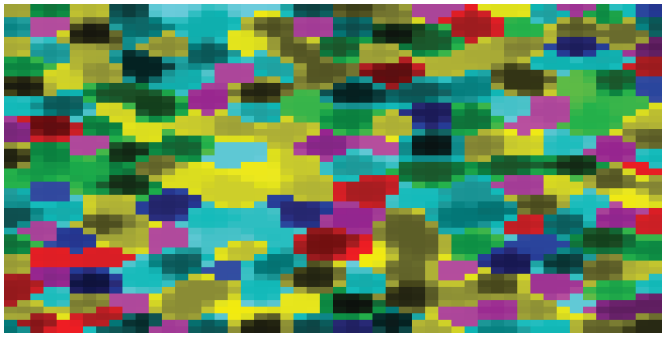


Figure 2.4: The dominant emotion visualisation, where darker zones represent stronger emotion. Colour legend: Anger, Disgust, Fear, Joy, Sadness and Surprise

Our implementation of the SOM model does not use a single label representation. Instead, we have 6 different values, each corresponding to a different emotion. Using only a dominant emotion representation, we are able to show only the convergence of the most intense value. Figure 2.5 shows an alternative approach. We chose to represent, for each individual in our SOM configuration, all the valences that are close to the dominant valence. This is done by selecting all the valences that are higher than 90% of the dominant valence. This allowed us to create a new visualisation, where larger continuous areas represent more intense emotions.

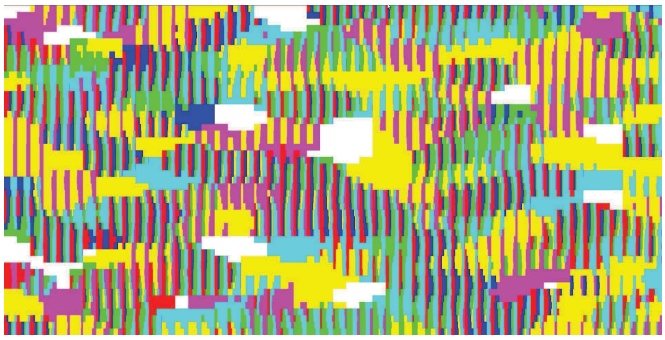


Figure 2.5: The top dominant emotions visualisation, where larger areas of the same color represent more intense emotions. Colour legend: Anger, Disgust, Fear, Joy, Sadness, Surprise and white represents No Emotion

2.5.5 Results

During the SemEval 2007 task, the coarse-grained evaluation did not provide good results. Therefore, we started with two experiments in order to discover any kind of class dominance. Firstly, only the emotional values were taken into consideration, but this approach failed to extract any dominant class. Secondly, the neutral class (No Emotion) was added, leading to an important result, as shown in Table 2.2. The neutral

class is observed with a strong dominance over the other classes, i.e. 64 % dominant value. The conclusion of this experiment is that neither of the classifiers presented in the SemEval 2007 conference managed to break the dominance of the neutral class, and the classifier we proposed discovers the neutral class better than the others, as seen later in our experiments.

Emotion	Nb. of instances	Percent
No emotion	642	64.85%
Anger	14	1.41%
Disgust	6	0.61%
Fear	65	6.57%
Joy	110	11.11%
Sadness	81	8.18%
Surprise	38	3.84%
Combined	34	3.43%

Table 2.2: Dominant class for coarse-grained representation

The second experiment we conducted concerns the whole corpus, with a coarse-grained representation, like the one described in Section 2.5. All the results are presented in Table 2.3. The LSA training column represents the LSA decomposition method applied on the words extracted from the training corpus, whereas the LSA Gutenberg column presents the results of the k-LSA method applied on the 10 000 words extracted from the Gutenberg corpus, as described in the Section 2.5.

Emotion	LSA training			LSA Gutenberg		
	Prec.	Rec.	F1	Prec.	Rec.	F1
Anger	10.00	11.86	10.85	**18.52**	15.38	**16.80**
Disgust	3.33	4.17	3.70	**8.33**	7.69	**8.00**
Fear	19.01	17.76	18.36	28.39	27.67	**28.03**
Joy	36.75	36.75	36.75	40.49	64.62	**49.79**
Sadness	24.14	40.00	30.11	27.08	19.60	22.74
Surprise	**29.73**	6.92	11.23	22.50	4.95	8.11

Table 2.3: Results for each emotional class

In order to evaluate our results, we also present the most significant scores obtained by the systems participating in the SemEval 2007, task 14 competition [187], in Table 2.4. The LSA All emotional system [187], is a meta-LSA method applied on the corpus, using as support words those existing in the WordNet Affect database and all direct synonyms, linked by the synset relation. UA [107] uses statistics gathered from three search engines (MyWay, AlltheWeb and Yahoo) to determine the amount of emotion in each headline. Emotional score is obtained with the Pointwise Mutual Information

Emotion	LSA All emotional			UA			UPAR7		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Anger	6.20	**88.33**	11.59	12.74	21.60	16.03	16.67	1.66	3.02
Disgust	1.98	**94.12**	3.88	0.00	0.00	-	0.00	0.00	-
Fear	12.55	**86.44**	21.92	16.23	26.27	20.06	**33.33**	2.54	4.72
Joy	18.60	**90.00**	30.83	40.00	2.22	4.21	**54.54**	6.66	11.87
Sadness	11.69	**87.16**	20.62	25.00	0.91	1.76	**48.97**	22.02	**30.38**
Surprise	7.62	**95.31**	14.11	13.70	16.56	**14.99**	12.12	1.25	2.27

Table 2.4: The systems presented in the SemEval competition

(PMI) algorithm. UPAR7 [35] is a rule-based system with a linguistic approach. The system uses the Stanford syntactic parser on the titles and identifies information about the main subject by exploiting the dependency graph obtained by the parser.

The overall results of all presented systems are described in Table 2.5. The global performance of the system is evaluated by making a strict overlap between the desired emotion vector and the obtained one, in the coarse-grained context.

Model	Precision	Recall	F1
LSA training	20.50	19.57	20.02
LSA Gutenberg	24.22	23.31	**23.76**
LSA All emotion	9.77	**90.22**	17.63
UA	17.94	11.26	13.84
UPAR7	**27.60**	5.68	9.42

Table 2.5: Overall results

The results are not surprising, because the LSA All emotions offers a good coverage over the emotional words, but its synonym expansion algorithm introduces a lot of noise in the method, and therefore offers a very poor precision. UPAR7 leads in some cases to a better precision, due to its analytical nature, but it lacks in recall. Our system is a good compromise between precision and recall, as F1 measure shows.

Unfortunately this approach needs a large collection of documents, as Project Gutenberg, in order to create the support dictionary for the SOM. The linguistic model compiled using Project Gutenberg is not very suitable for modern language description, since the repository contains books written in old English. Moreover, the SOM could be replaced by more efficient machine learning techniques (i.e. SVM). Given so, we decided to adopt methods that are more linked with the dialogue problem.

2.6 The dialogue characteristics

In interactive systems dealing with spoken language and more particularly in affective dialogue systems, the transcribed part of the oral interaction is not as rich as for other

environments, such as blogs or newspaper articles. In most of the cases, the transcription provided by a Speech-to-Text or Automatic Speech Recognition engine, offers only the text information, without any punctuation, emoticons or pause clues. The current transcribers provide a poor quality for open context dialogue, and the speed is usually not real-time. Moreover, the detection rate is decreased by presence of homophones, different accents or even mixed languages. All these issues related to the transcription technology make the valence/emotion detection problem even more difficult.

Most of the classic systems built for valence extraction [134] use acronyms, emoticons, most common spelling errors and "internet slang", among other classic linguistic features. Unfortunately, none of these approaches help when dealing with transcribed data. Nevertheless, the systems that are dealing with spoken language have to take most of these errors into account. Otherwise, the system would not take accurate decisions, therefore multiple modalities have to be considered.

Because of transcription errors, a system dealing only with text data would not have sufficient information, the text may have missing words or other phonetic artefacts related to the transcriber. Homophones (i.e. words that sound alike with different meaning or spelling) are one of the big problems of transcription. Some very common examples are: *bear* (to bear something:verb) and *bear* (animal:noun); *accept* and *except*; *altar* and *alter* or *ate* and *eight*. Transcribing the wrong meaning of a word could severely decrease the precision of a system. To prevent this from happening, some systems, offer multiple transcription possibilities, ranked by their confidence ratio.

Apart from text, the transcription can be enriched with features from multiple modalities, such as acoustic features or visual to improve valence/emotion detection [73]. Multi-modal systems rely on multiple sources of information, considering the conversational partner and the interaction with him. The usual modalities considered for this task are voice qualitative features (e.q. pitch, power, volume, pauses), voice transcription and video features (e.q. gestures, postures, gazing, smiles), as presented in Calvo et al. [28].

As the smile is usually associated to emotional context [126], we compare several feature representations for a valence detection model that fusions the presence/absence of smile as visual feature and the text transcription. Other gestures, such as eye gazing or head movement, are linked more with activation or power than with valence [73].

2.6.1 Youtube Opinion Corpus

The various feature representations we propose are evaluated on the Youtube Corpus of Morency et al. [126] which consists in a series of video blog extracts. Each individual expresses an opinion towards a product, person or event. This chapter focuses on opinions expressed by users in a video blog scenario (monologues). The chosen monologues consists in transcribed text log, as the result of verbal interaction. Moreover, it contains a series of visual and vocal features corresponding to several gestures and instant vocalisations. These logs are annotated with the identity of the speaker. Usually, the transcription does not use the same internet slang nor emoticons as a chat system does.

In comparison to the previous work done on this corpus, we introduce the idea of a segmented detection for text data rather than predicting a single label for the whole video. Moreover, we study the importance of several features in the process of multi-modal fusion, such as NGrams, smile effect or words carrying affective valence.

The corpus chosen for our approaches is the Youtube Opinion Corpus proposed by Morency et al. [126]. This contains a selection of 48 videos selected from the Youtube.com website. All these videos have been automatically transcribed and provided with a couple of annotated features: audio features (extracted with OpenSmile [60]) and video features (extracted with *OKAO*, CLMZ, GAVAM or SHORE). More technical details are available in the corpus documentation [126]. For our approach, we use only the smile feature extracted with OKAO Software.

OpenSmile

The openSMILE feature extraction tool allows the extraction of real-time large audio feature spaces. It combines features from Music Information Retrieval and Speech Processing. SMILE is an acronym for Speech & Music Interpretation by Large-space Extraction [60]. All these features have been extracted in Weka ARFF format [74], which allows an easy usage in many statistical machine learning toolkits.

OKAO

The OKAO, which means *face* in Japanese, features have been extracted with OKAO Vision Software[4] by the authors of the corpus. It provides information about eyes gazing, head movement and activity, lip motion and smile intensity. Whereas the features linked with eye and head activity are correlated with energy and arousal, the lip activity and smile can be associated with valence, as the smile tends to be associated with positive expressions [126]. Therefore, the smile intensity is one of the most representative face features extracted with the OKAO toolkit.

Valence Annotation

The annotation is done by three different annotators, and consists in three valence labels (-1 for negative, 0 for neutral and +1 for positive, presented in the first column of Table 2.6 as well) associated to the data expressed by the user present in the Youtube video. The annotation was conducted on all the videos, splitting them into multiple segments. The segments were decided independently by each annotator and the final ones were decided by voting the overlapping ones. The final label for each segment is decided by a majority vote. In case of an indecision, a neutral label is chosen.

Table 2.6 provides an example of data from the Youtube corpus. Each line from the table corresponds to an annotation segment. The transcription may have overlaps with the previous or next segments, because the transcription was contained in multiple annotation segments.

[4] `http://www.omron.com/r_d/coretech/vision/okao.html`

Annotation	Smile	Transcription
-1	Yes	*i can not stand it*
-1	Yes	*i can not stand it people are f*** retarded*
0	Yes	*people are f*** retarded like i was handing this lady her soda is one was a doctor pepper or something one was a sprite*
0	No	*one was a doctor pepper or something one was a sprite she looks at me she looks at the sodas and was like which one is the sprite*
-1	No	*which one is the sprite lady okay if you are colorblind that is cool but are you retar like are you serious quit playin*
-1	Yes	*are you retar like are you serious quit playin are you that stupid whatever*
-1	Yes	*are you that stupid whatever i am like i jus i have to deal w like i mean i am a people person i love people i you know i love talking with people it is just*

Table 2.6: An example of transcription and smile presence from Youtube Database, video 7. The transcriptions and smiles are provided per segment basis. The bad words are censored with *

Starting from this corpus, we present our methodology which is used to generate our features for the fusion model.

2.7 Multi-modal Affect Detection

Our methodology uses two modalities to detect human emotions: speech and gestures. In order to apply semantic analysis techniques, the speech is transcribed into text. For the gesture modality, we chose to evaluate the smile since it is usually associated with affective feedback. Other gestures, such as eye gazing or head movement, are linked more with activation or power than with valence [73].

Another very important aspect of any emotion detection system during dialogue interaction is represented by the functional segment, from which the roles are modelled. These segments can be a series of frames, when they are used to predict affects based on video or acoustic features. For text applications, the segments could be the words, the utterances or other functional structures. In a more general case, the frames can be grouped into a series of functional segments, based on their role on the video.

In figure 2.6, the rounded orange boxes represent the gesture segments, the blue squared box represent the transcription and the dark arrows labelled as s_* represent the annotation segments. For the previous example, the non-strict transcription overlapping means that the first blue text segment (the lower box), is considered in all the s_i, s_{i+1} and s_{i+2} segments. On the visual level, gestures are characterised usually by type, intensity and duration. This kind of information can be described by the annotation segments. Whereas the type of the gesture remains the same all over the segment, the

intensity and duration is adjusted by the length of a certain annotation segment. In figure 2.6, the first gesture (first orange rounded box) would be split into three different segments, all having different durations and intensities for segments (s_{i-1}, s_i and s_{i+1}).

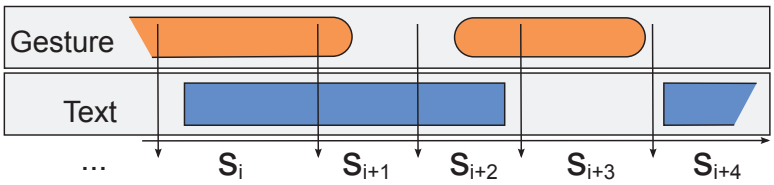


Figure 2.6: A multi-modal representation of the text and gesture interaction, with a segmented annotation. The segments s_i are chosen by the annotator based on their function, directly on the video sequence, without strictly overlapping the gesture or text track

Usually, the segments are chosen by the annotators based on their semantic function and could overlap the transcription or gesture segments. For the strict overlap, in general this is not the case, as the annotation could be done directly on the video data, without taking into account the transcription or gesture segments. Usually, the transcription and annotation segments do not strictly overlap and the transcription could not be segmented very accurately when the annotation segments do not include it. In this scenario, since the semantic relations between words are very important, we therefore decided that the transcribed segments are taken as a whole for each annotation segment, even if the transcription time frame is passing the annotation bounds.

All these presented concepts need to be formalised and encoded into numerical features. Afterwards, they are provided to a classifier, which, in combination with our features, constitutes our classification model.

2.7.1 Classification Model

Similarly to the SOM model presented before, our technique requires a multi-step process. The first step, also called preprocessing, consists in the filtering and cleaning the text information. The feature extraction and a projection follows, by using simple word presence strategies. In the third step, the SVM optimisation is applied and the trained model is used in the classification step.

We present first the SVM classification algorithm and we continue with the description of the training and testing sets. We follow with the feature extraction process and a brief description of the results.

2.7.2 SVM Classifier

The classifier we have chosen is a commonly used supervised method, the Support Vector Machine (SVM). We use the implementation provided by the Weka toolkit [74], which offers a set of classification kernels, such as linear, polynomial or RBF. We also used the SMO [100] implementation for the SVM optimisation. SMO offers the possibility to configure multiple kernels and we have chosen the linear kernel, using the balance criteria between the training speed and accuracy of the results.

In the case of an SVM, we define the data samples as $\mathcal{X} = \{x_k | x_k \in \mathbb{R}^n\}$, and the classes (labels) as $\mathcal{Y} = \{y_k | y_k \in \{-1, +1\}\}$. The solution for the SVM problem is a function, $f(x), \forall x \in \mathbb{R}^n$, that separates the space ($\mathcal{X}$) into two hyperplanes. Ideally, all the samples labelled with $+1$ are contained in one hyperplane and the samples labelled with -1 are in the other.

The mathematical definition of the function, for the linear case, is given by the following equation:

$$f(x) = \langle w, x \rangle + b = x^T w + b \tag{2.10}$$

where the $\langle a, b \rangle$ represents the *dot product* of a and b.

Furthermore, finding this equation is a problem of "Quadratic Programming Optimisation" of a linear function under constraints. This could be solved in several ways, starting from the mathematical form of the problem:

$$\min_{w,b} \sum_{i=1}^{N} [1 - y_i \underbrace{(\langle w, x_i \rangle + b)}_{f(x_i)}]_+ + \lambda \|w\|^2 \tag{2.11}$$

where N is the number of samples used to train the kernel, λ is a penalty coefficient used to prevent the function overfitting, and $\| \bullet \|$ is the Euclidean Norm.

For the context of this experiment, we use the SMO [100] algorithm for the SVM optimisation, which implements John C. Platt's sequential minimal optimization strategy [153], provided by Weka Toolkit.

Another aspect of the SVM problem is represented by the "margin". It is the intersection of two hyperplanes, define by the following equations: $\langle w, x \rangle + b = -1$ and $\langle w, x \rangle + b = +1$. Figure 2.7 shows an SVM representation by using a margin. Its role is to find the best function that produces a linear separation of the plane, while minimizing the margin:

$$\min_{w,b} \frac{2}{\|w\|} \tag{2.12}$$

To obtain the class after the function is computed, the sign of the function needs to be evaluated: if $f(x) < 0$ then the sample (x) is in the -1 class, and in the $+1$ otherwise.

2.7.3 Training and testing sets

Since the corpus does not provide a training and testing set, we separate it, based on the user-independence strategy. Therefore, the split has to be done in a way that a subject in the training is excluded from the testing.

The strategy we propose is a 47-fold evaluation, where the training is done on 46 videos and the testing on 1. This method has been proven to be useful to discover outlier users (users that do not fit the training data). Morency et al. [126] used a similar method to test their approach, the only difference was that they used 48 folds, because they did not eliminate the video number 20 from the dataset. For the purpose

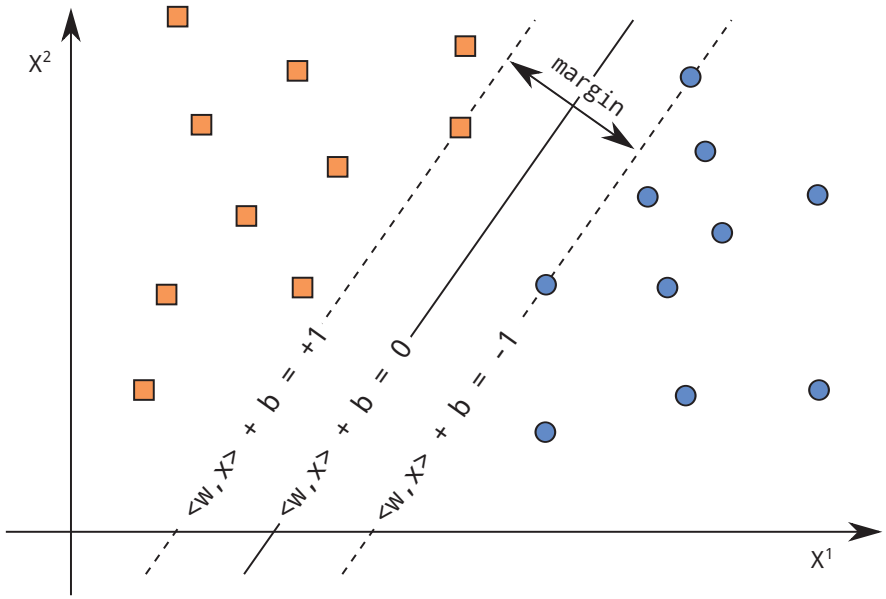


Figure 2.7: The SVM hyperplane separation using a linear kernel and margin equations

of our experiment, this video was removed from the original corpus since it does not contain any OKAO annotation, and therefore no smile level could be extracted.

Since SVM is a binary classifier, we split the classification method into two different approaches: a neutral vs non-neutral classification (positive vs negative classifier) and a positive vs non-positive (neutral vs negative). The separations are done as a two level classification, we apply the first level of classification and for the combined class, we apply a second level to discriminate once more. In the case of neutral vs non-neutral, this means that at first we discriminate the two classes with an SVM kernel, and next, in the case of the non-neutral class, we apply a second classifier to discriminate the positive against the negative class.

The choice of a two level SVM has been made because we want a more detailed analysis on the feature fusion mechanism.

2.7.4 Feature extraction

Preprocessing Step

During the preprocessing step, on each transcription segment of each video, a collection of filters is applied, in order to remove any useless information, such as meta-information for pauses, which is specific to every transcriber. The short word structures are expanded to their full representation, such as *don'* (*don't*) which is expanded to *do not*. Concerning the final feature extraction algorithm, any partial transcribed words are filtered, such as *stu- stupid*, in which case we filter the word *stu-*.

In order to increase the precision of our system, we apply a Part Of Speech Tagging (POS). The dictionaries we use offer valence only for nouns, verbs and adjectives. Moreover, some very frequent English words may have the role of prepositions (i.e. *like*). The POS tagger used is the one of SENNA [39], which offers both good precision and speed.

Feature Extraction

In order to proceed to our SVM classification, we extracted a set of feature vectors.

NGrams are one of the few basic word features used in text understanding. Given a window of length $n = 1, 2, 3...$, we consider all the word combinations in the given window. These groups are unordered. All the NGrams are first extracted from the training corpus and a dictionary ($\mathcal{D}^{NGram}$) is compiled, with all the possibilities found in the corpus.

Based on the dictionary already compiled, a second pass is done, and a NGram word presence vector is computed. This means that $\forall w_i \in \mathcal{D}^{NGram}$, the i of the vector is 1, if the word from the $\mathcal{D}^{NGram}$ is contained into the sentence, as shown in the equation 2.13.

$$\forall w_i \in \mathcal{D}^{NGram}, v^{NGram}(w_i) = \begin{cases} 1 & w_i \in sentence \\ 0 & w_i \notin sentence \end{cases} \tag{2.13}$$

1Gram is a specific type of NGrams, for the case where $n = 1$. In this scenario, the dictionaries $\mathcal{D}^1$ are generated as well as the associated feature vectors.

Smile feature has been extracted using the given OKAO video features. OKAO provides for each frame of a video an intensity for the smile level, which has to be filtered because of the noise. The smoothing is done with a window average filter, which takes a window of 50 frames and calculates an average for the current frame. Secondly, a threshold has been applied to detect the peeks into the continuous signal, as it can be seen in figure 2.8. These peeks correspond to clear smile segments.

As the smile is user dependent, detecting it with a high precision is very difficult. Therefore, we applied three different thresholds to segment the smile. Based on the selected thresholds, the 40% intensity is a very optimistic estimation of smile presence and the 60% is very pessimistic and detects only the clear smiles. The 50% level offers a good balance between the previous two thresholds, but it does not cover all the individual smile types.

The smile feature is computed as a single presence, very similar to the NGram vectors. The dictionary for the smile vector consists in the three smile-based features, segmented by the thresholds presented.

AVW (the Average of Valence given by Words) feature has been computed as an average of the positive and negative percent, given by all the words contained in the selected segment. The valences are given by SentiWordNet [11]. This features was designed to let the SVM take the final choice from the two valences computed on the word level. In future, this feature could be replaced by a contextualised word valence, presented in **Chapter 3**.

1Grams + 2Grams Given the efficiency of the 1Grams, we merged the two features together as a simple vector merge and feed the resulting vector to the SVM. The 2Grams

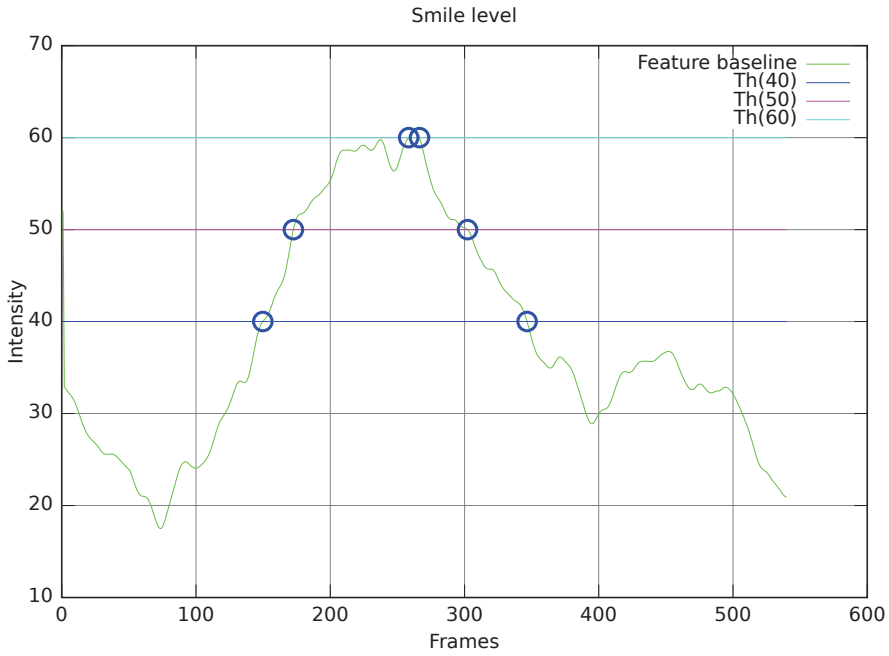


Figure 2.8: The continuous smile function along with three different thresholds: 40%, 50% and 60% according to most common smile levels in the corpus. The signal has been already smoothed with an average windowed filter. The blue circles represent the intersection point between the signal and the threshold line.

are similar to 1Grams and have the $\mathcal{D}^2$ dictionary associated.

1Gram + Smile This is a feature merge between the 1Gram and Smile feature vector. It has been chosen based on the good results obtained for each feature independently.

1Gram + S-W The **Smile-Word (S-W)** co-occurrence is a feature built to detect segments where the words (1Grams) are influenced by the smile presence. The dictionary for the feature vector is built in a similar way to the NGrams, the only restriction is that the smile and the vectors should occur on the same segment. If a smile is present into a segment, it is considered that it influences all the functional words present in that segment. This is different from the 1Gram and Smile feature merge because of the co-occurrence strategy considered.

1Gram + S-All This is a feature merge between 1Gram, Smile and Smile-Word. It has the role to combine all the features that consider smile along with the semantic features corresponding to 1Grams.

1Gram + AVW feature merge is a feature merge between the 1Gram and AVW feature vector.

2.7.5 Results

The results are presented for the two separations we considered. Our classifiers are built for a two level decision: First Level is a Neutral vs Non-Neutral discrimination, and in

the case of a Non-Neutral level, a Second Level classifier is done to discriminate positive vs negative. These results are presented in the Table 2.7. The levels considered for a second separations are similar to the first ones: Positive vs Non-Positive and Neutral vs Negative. These results are presented in Table 2.8.

	First Level			Second Level		
	Neutral vs Non-Neutral (+/-)			Positive (+) vs Negative (-)		
Model	Prec.	Recall	F1	Prec.	Recall	F1
1Gram	0.6449	0.5946	0.6187	0.7147	0.5176	0.6004
Smile	0.5759	0.6813	0.6242	0.4909	0.5502	0.5189
AVW	0.5544	0.7061	0.6211	0.4075	0.5006	0.4493
1Gram + 2Gram	0.6080	0.6421	**0.6246**	0.7095	0.5275	0.6051
1Gram + Smile	0.6734	0.5853	**0.6263**	0.7285	0.5384	**0.6192**
1Gram + S-W	0.6272	0.5750	0.6000	0.7085	0.5027	0.5881
1Gram + S-All	0.6452	0.5849	0.6136	0.6926	0.4902	0.5741
1Gram + AVW	0.6370	0.5845	0.6096	0.7194	0.5536	**0.6257**

Table 2.7: Results for the neutral vs non-neutral separation for the first level of classification and positive vs negative for the second level

In case of the first experiment (Table 2.7), the best features for the first level are 1Gram+2Grams and 1Gram-Smile. Out of these two, 1Gram+2Grams has a better recall, while 1Gram-Smile is more precise. On the positive vs negative discrimination, the 1Gram+Smile and 1Gram+AVW seems to offer the best results, both of them having good precision, as well.

	First Level			Second Level		
	Positive vs Non-Positive (0/-)			Neutral (0) vs Negative (-)		
Model	Prec.	Recall	F1	Prec.	Recall	F1
1Gram	0.6055	0.6192	0.6123	0.6307	0.5353	0.5791
Smile	0.5591	0.6846	0.6155	0.4857	0.5090	0.4971
AVW	0.5421	0.6945	0.6089	0.4168	0.5125	0.4597
1Gram + 2Gram	0.6125	0.6712	**0.6405**	0.5990	0.5275	0.5609
1Gram + Smile	0.6198	0.6213	0.6206	0.7146	0.5575	**0.6264**
1Gram + S-W	0.6295	0.6289	0.6292	0.6428	0.5201	0.5750
1Gram + S-All	0.6485	0.6369	**0.6427**	0.7184	0.5516	**0.6240**
1Gram + AVW	0.6146	0.6160	0.6153	0.6489	0.5279	0.5822

Table 2.8: Results for the positive vs non-positive separation for the first level of classification and neutral vs negative for the second level

Table 2.8 presents the results for the second separation considered. For the first

level, 1Gram+2Gram and 1Gram+S-All offers the best results. For the second level, 1Gram+Smile and 1Gram+S-All give the best results.

Out of these two experiments, it seems more easy to discriminate positive information by using a smile, on the first level. Moreover, the smile used as a feature seems to be very useful to discriminate affective information.

The AVW feature, which is used in most of the dictionary based approaches, seems to work well in combination with 1Gram features, to discriminate between positive and negative. Nevertheless, this approach computes first the valence as a feature, which is used by the SVM to discriminate the valence classes.

On the original Youtube dataset, Morency et al. [126] used a simple affective word presence feature (Text only), smile and eye gazing time (Visual only), pitch and power (Audio Only). For the classification, they used a three class HMM-based classifier and for the fusion, a simple feature merge was used. The best result is obtained for the fusion. All the results obtained by Morency et al. [126] are presented in Table 2.9. In comparison with these results, we managed to obtain better ones by using only text and visual modalities.

Modality	Precision	Recall	F1
Text only HMM	0.431	0.430	0.430
Visual only HMM	0.449	0.430	0.439
Audio only HMM	0.408	0.429	0.419
Tri-modal HMM	0.543	0.564	**0.553**

Table 2.9: Detection results for a three class HMM-based classifier, as extracted from Morency et al. [126]

2.8 Discussion

For our first experiment, on the SemEval 2007 corpus [187], the results prove our initial goal: to design an algorithm that is fast and would provide a good equilibrium between precision and recall on text data. The choice of a strategy, from the ones presented on the SemEval 2007 competition, for an emotion detection task is a matter of performance required by the application: in an offline environment[5], having a semi-supervised filtering task, the LSA All emotion and UPAR7 would probably offer good results, but in an online processing task, our strategy performs better, while offering a good balance of precision and recall.

Even if this approach needs a large collection of documents, as Project Gutenberg, in order to create the support dictionary for the SOM, it could be used in a combination with other dictionaries for more formal writing styles (i.e. blogs, twitter).

The second experiment is designed in a multi-modal context, by taking into account

[5]In the concluding remarks of the Strapparava et al. [187] paper it is said that several strategies, including UPAR7, perform very poor in terms of speed

semantic information and smile. The smile feature can be used to boost the results, for both of the classification problems neutral vs non-neutral and positive vs negative. Whereas this feature is believed to be linked with strong affective content, it can be used to discriminate the positive data as well.

The AVW, one of the most widely used text feature in Affective Computing, does not seem to discriminate well the valences, even in the neutral vs non-neutral separation. The interesting fact observed on this feature is that a combination of positive and negative words, as extracted from a dictionary, does not necessarily produce a negative or a positive phrase, only by counting the valence. This would suggest more complex approaches, such as AFFIMO, proposed by Ochs et al. [134], or by using contextualised affective dictionaries, as we propose in the next chapter.

The classifiers used for the two experiments are different and adapted for the task. The SemEval 2007 corpus provides a continuous annotation for each emotional label, whereas the Youtube Corpus uses discrete valence labels. For the first task, the SOM is used to overcome the fuzziness of the annotation process. Based on the results obtained, this strategy proves to be efficient. The SVM, which is a Supervised Machine Learning technique, uses the annotation to learn the classifier. Moreover, it suits well discrete predictions that are made on a window level.

In the current approaches, we used WordNet Affect, which attaches to each word the most common emotional label and SentiWordNet, which has many conflictual valences for the same word. One of the solutions for this problem is to attach to each word a context and for each context an affective label. By using this, the number of semantic conflicts will decrease. Our next proposition deals with this problem by automatically generating an affective context graph.

CHAPTER 3

Affective linguistic resources for emotion detection

"Human behaviour flows from three main sources: desire, emotion, and knowledge."

– Plato

Contents

3.1 Context 52
3.2 Related work 53
3.2.1 Affective Dictionaries 53
3.2.2 Maximal Clique Exploration 54
3.3 DDMCE Algorithm 59
3.3.1 Preliminaries 59
3.3.2 Algorithm 61
3.3.3 Implementation 65
3.3.4 Experiments 68
3.4 A new linguistic resource: affective contextonyms 71
3.4.1 SentiWordNet 71
3.4.2 Modelling context with contextonyms 74
3.4.3 A new linguistic resource: an affective contextonym model for dialogues 76
3.4.4 Validation 79
3.5 Discussion 81

3.1 Context

In the field of sentiment analysis and emotion detection based on text data, two main research directions exist: 1) building large and reliable linguistic resources, such as dictionaries or ontologies and 2) designing classifiers for valence, sentiment or emotion detection [28]. However, since affective classifiers usually rely on linguistic resources, these dictionaries have to be large and accurate.

Unfortunately, even if recent approaches have increased the size of the dictionaries, the ambiguity of the decision increased as well [156]. Our goal in this chapter is to improve these dictionaries by preserving, as much as possible, the annotation accuracy. This objective is performed by taking into account the context of the word, and a new linguistic approach to model this relation, called the contextonym model. This process is done through multiple phases: a large collection of documents is selected (in this case a subtitles corpus), afterwards a clique exploration algorithm is considered to generate the contextonym model. Over this model, the SentiWordNet (SWN) [11] valences are mapped and in certain cases disambiguated. The innovative characters of this approach consist in using subtitles to model a dialogue context, the clique exploration algorithm proposed and the automatic valence disambiguation using a context-graph.

The corpus considered for this experiment consists in a snapshot of a subtitle database, obtained from `http://www.opensubtitles.org/` and `http://www.podnapisi.net/`. This database has been chosen because the documents are available in large quantities, multiple languages, their quality can be assessed using simple measures and they are free of charge. Moreover, the information is expressed in an oral style, with a structure similar to dialogue.

This chapter also addresses the problem of finding maximal cliques in large dynamic graphs with an algorithm based on an original graph-to-tree transformation. When dealing with large and dynamic data, one of the solutions is to design distributed algorithms in order to reduce the computational time. In the context of distributing data over a network of processors, the optimal solution is the one that does not require all the data to be transported on all the machines on the network, but just a sub-image of the original data. We call this problem Graph Dimensionality Reduction. We, therefore, propose an efficient MCE algorithm called DDMCE (Dynamic Distributable Maximal Clique Exploration) that finds all the maximal cliques in a large dynamic graph, even when the original graph structure is modified while the algorithm is processing the remaining nodes. DDMCE is based on a graph to tree transformation, allowing to reduce the computation time, to deal with dynamic changes over a distributed architecture. As the size of the data cannot be reduced, no theoretical dimensional limit is imposed to the algorithm.

In this chapter, we present a brief state of art concerning different approaches used for emotion detection based on dictionary approaches and Maximal Clique Exploration. The DDMCE Algorithm is introduced afterwards with several experiments to validate the approach. We continue with the presentation of an automatic generated contextualised affective dictionary and we conclude with a brief discussion.

3.2 Related work

3.2.1 Affective Dictionaries

Very often, building a classifier, relies on having large and accurate linguistic resources. Improving these dictionaries, by preserving their size and increasing the annotation accuracy, is therefore considered mandatory.

Among the existing linguistic resources, WordNet (WN) [62], or variations over it, remains the most popular one [28]. WN, built at Princeton University, is used in most of the Natural Language Processing applications. The concepts in WN are grouped into synonym sets (also called synsets), which are sets of words semantically linked. Each synset description contains its frequency in the dictionary and a glossary which is basically a short sentence describing the synset. Among the basic synonymic relations, WN also contains some special relations such as hyponymy, hyperonymy or ISA ("is a"). All these links describe generalisation, specialisation or equivalence relationships between synsets.

As a synset database example, WordNet Affect (WNA) [188] is an extension of WN. WNA contains synsets annotated with emotional labels (i.e. Ekman's basic annotation scheme [59]: Anger, Disgust, Fear, Happiness, Sadness and Surprise). WNA contains nouns, adjectives, adverbs and some verbs for the English WN 2.0 version. ConceptNet [113] is another well-known ontology widely used for semantic disambiguation in classification tasks. This database contains assertions of common-sense knowledge encompassing the spatial, physical, social, temporal, and psychological aspects of everyday life. ConceptNet was generated automatically from the Open Mind Common Sense Project [180]. Finally, SentiWordNet (SWN) [11] is dedicated to opinion and valence classification. Valence is represented by the degree of positivity, negativity or objectivity of a certain word or sentence, whereas opinion represents the general valence over a series of sentences. SWN is the result of a semantic propagation algorithm over all WN synsets according to their valence.

All these linguistic resources are used, among other applications, to design affective classifiers. For instance, starting from WNA, Valitutti et al. [198] proposed a simple word presence method in order to detect emotions, where the emotion of a sentence is given by the dominant affective word. SemEval 2007 (task 14) [187] presented a corpus and some methods to evaluate it, based on WNA as dictionary [187]. In particular, methods based on a fusion of algorithms, using Keyword Spotting, Lexical Affinity or Statistical Natural Language Processing are very popular [134]. All of them use certain affective dictionaries to train their methods: 1) WNA [188]: EmoText [139], EMMA (Emotion Metaphor and Affect) [209], EmoHeart [131]; 2) Mind Common Sense [180]: EmpatyBuddy [114]; 3) SWN [11]: AFFIMO [134]; 4) Hand crafted rules and dictionaries: Emologus [3], Sentistrength [192].

All the linguistic resources used in these methods are either accurate small scale dictionaries or large ones with a high number of conflicts. These are defined as valence inconsistencies or ambiguities which appear for words contained in the same linguistic

resource. More details about this will be given in the following sections.

The larger the support dictionary is, the higher recall the corresponding system obtains. On the contrary, the less conflicts the linguistic resource contains, the higher the precision of the classifier is. Increasing both the size and annotation accuracy of linguistic resources is therefore considered necessary. Unfortunately, even if recent approaches have increased the size of the dictionaries, the accuracy of these methods still decreases [156].

Our goal is to improve these dictionaries by preserving, as much as possible, the annotation accuracy. This could be done by introducing context into existing large dictionaries, such as the SWN, which would decrease the number of conflicts. This objective is performed by taking into account the context of the word, and a new linguistic approach to model this relation, called the contextonym model. The contextonyms are modelled as maximal cliques in a word co-occurrence graph.

The algorithm used to build the contextonyms graphs needs to be able to explore large and dynamic data. Moreover, due to a high computation time, this process needs to be done in a parallel or distributed way.

3.2.2 Maximal Clique Exploration

A clique is a complete sub-graph, i.e. a sub-graph such that every two nodes are connected. A clique C is maximal if no larger clique contains C. Semantically, a clique contains all the elements that are somehow strongly related, based on a specific function, such as contextonyms.

A graph can contain multiple cliques and the number of cliques cannot be approximated by any simple mathematical formula. Usually, it depends on the number of nodes and the density of the graph. Clique discovery algorithms can be classified in three different categories: Clique Counting (CC), Largest Maximal Clique Exploration (LMCE) and Maximal Clique Exploration (MCE). CC algorithms return only the exact number of cliques detected, rather than a list of them, which limits dramatically their domains of application. LMCE, addressed early in 1996 during the DIMACS Second Challenge proposed by [96], is concerned with discovering only the largest clique, rather than exploring all the candidates, as for MCE.

Clique discovery in large graphs is a scientific issue that started in the early days ([115]) of the modern computer science. It is shared by various fields such as computer vision (i.e. feature matching [20], scene labelling [36, 76, 125, 164], stereo matching [120, 130, 34, 80]), information retrieval (i.e. document alignment, similarity measure [23, 61, 25], document indexing [119, 69]), computer network task planning [50], protein segmentation in biochemistry problems [169, 104, 146, 95], construction of linguistic resources for machine translation or dialogue analysis (i.e. extraction of context cliques called "contextonyms" on large graph-based linguistic models [88, 121]) and more generally in all the problems where efficient clique searching algorithms are needed.

More recently, MCE algorithms have been applied to modern social network analysis [85, 124], where the clique could represent a close group of friends of an individual. The

techniques used in this area are often similar to the bio-chemistry related algorithms, and usually only the interpretation of the cliques differs. This kind of application raises new challenges since the data to analyse are dynamic: whereas for classic applications the graph is static in most of the cases, the structure of social networks mutates with a high frequency. Classic MCE algorithms need therefore to be adapted in order to compute a clique discovery not from scratch, but from a previously computed exploration, which can also be partially up-to-date.

In all these applications, MCE algorithms are considered as extremely challenging, because of their NP-hard status [99]. Exploring all the candidates in a graph ensure to discover the optimal set of maximal cliques, but is time consuming because no early prediction of "false paths" can be made. For certain problems, when a complete list of all the existing cliques is not needed, heuristic approaches sacrifice optimality and completeness to gain computational time. Furthermore, since a dynamic graph changes its structure, the time needed to take into account such a change is important. Nevertheless, choosing an optimal approach or a heuristic one is problem dependent. The application domain also influences the dynamicity of a graph and computational time required to process it.

Nowadays, proposing an efficient distributed approach for the MCE of large and dynamic data is still an open problem. Many research teams have proposed different strategies to reduce both the exploration space and the computational time, as detailed in Cazal's survey [32].

In the following of this section, we present the two main directions for the MCE: the exploratory algorithms for a complete exploration of a given graph space and the heuristic strategies allowing to partially explore this graph space. We also present the most significant implementations of MCE algorithms. The current method focuses on the exploratory approaches, since we consider important the optimality and completeness ensured by these algorithms. Nevertheless, considering large and dynamic data, optimizing the computation time, related mostly with the heuristic approaches, remains also one of the priorities of our research work.

Exploratory Clique Algorithms

The term "MCE " is used only in the context of full exploration. The main advantage of these algorithms is that completeness and correctiveness can be proved [24, 5, 103, 196, 32]. Bron & Kerbosch proposed in early 1973s one of the first MCE algorithms based on a depth-first approach to explore all the maximal cliques in a graph. Most of the algorithms used for MCE are descendants of this depth-first approach. Cazals et al. [32] separate MCE algorithms into two distinct classes: the "greedy" ones, as extensions of the algorithm proposed by Bron & Kerbosch [24] and Akkoyunlu [5]; and the output-sensitive ones, such as those proposed by Tsukiyama et al. [197] or [117]. The specificity of the exploratory algorithms is that, although the data representation differs from regular sets or matrices, all of them build a logic exploration tree, which is not represented in memory. This exploration logical tree has the same role as a function

call tree.

Bron & Kerbosch [24] provided a depth-first exploration algorithm (detailed in the next section as the Algorithm 3.3.1), which is still one of the most used algorithm for greedy search. The proposed exploration takes into account only the best option at each step, and does not apply any backtrack exploration. A variant of this algorithm published by Akkoyunlu [5] uses a similar logic exploration tree. Koch [103] published an improved version of the algorithm and introduce a pivot selection step, which consists in choosing a node that potentially reduces the exploration space. The idea of using a pivot selection strategy to improve the exploration was therefore often taken up; Cazals & Karande [32] studies the most recent strategies, and they concludes that the Tomita et al. [196]'s strategy is the most efficient one, for different exploratory algorithms. After the publication of this strategy, most of the MCE research has been focusing on implementing the method on different languages and software environments, rather than improving the algorithm itself. We will describe this algorithm in the next section.

The problem of clique exploration on dynamic data, has been firstly addressed by Stix [185], where the dynamical character of the graph is simulated by adding new edges. The method proposes several decompositions of the original graph into smaller structures, in order to reduce the algorithmic complexity and to scale better the addition of a new edge.

Exploring all the solutions in a graph is time consuming, mainly because no early prediction of false paths is possible. Heuristic approaches propose to improve the computational time by sacrificing the precision. These strategies could be applied for both large or dynamic data.

Heuristic Maximal Clique Algorithms

The Heuristic-based Maximal Clique (HMC) algorithms are usually not defined as exploration problems. A good review on the HMC problem can be found in Bonze et al. [21]. Among all the research directions for heuristic algorithms, three main classes can be distinguished: evolutionary approaches, dynamic local search and ant colony optimizations. Whereas the genetic algorithms propose a solution where an initial population is evolved to an optimum, the local dynamic search is an iterative approach, combining different strategies. The Ant Colony optimisation approach uses "pheromone trails" to reinforce certain paths in the graph, in order to build the cliques.

Following the observations made in [4], Balas et al. [12] propose a revised optimized crossover operator for an evolutionary approach, with standard genetic algorithm operators (selection and mutation) and maximal cliques represented as individuals.

Pullman & Hoos [160] presents a dynamic local search for the HMC approach, which combines two different strategies for building the cliques: a global iterative approach which has the role of adding new unexplored nodes to the clique, and a plateau search strategy which replaces nodes from the current studied clique with unexplored nodes. This algorithm seems to offer substantial computational time improvements over state of the art local search heuristic, as presented by Grosso et. al. survey [70].

Another popular heuristic approach is Ant Colony Optimization (ACO) [184, 71]. Solnon et al. propose a simple and efficient ACO algorithm, which generates cliques by successively adding new nodes into the clique set and uses "pheromone trails" as a greedy heuristic to choose the best node. The study also compares the two possibilities for laying the "pheromones": on the edge or at the node level.

The HMC algorithms obtain good results in all the application fields where reducing computational time is essential, rather than ensuring to collect the optimal complete solution. In practice, the choice of an HMC algorithm over an exact approach remains strictly linked to the application. More recently, with the development of Message Processing Interfaces, the implementations migrated from finding heuristics for the exploration problem to finding methods to process the data in a distributed way.

Distributed Systems

In the category of commercial implementations, the Google Pregel System [118] should be reminded, since it handles very large graph structures. The system has been built on the idea that every nodes of a graph could be activated or deactivated at any time. While active, a node can produce processing or activation messages. Malewicz et al. [118] present their architecture and the main capabilities of the platform, which includes: message passing, topology mutations and fault tolerance in large and dynamic graph structures. From the application perspective, some implementations are presented: Shortest path, Bipartite Matching (finding bipartite graph structures) and Semi-Clustering (an approach to find graph structures with strong links between nodes, while eliminating the soft ones). Even if the platform does not directly cover clique discovery, it remains one of the largest graph processing platforms. Unfortunately, the platform needs the whole data to be present on all the processing nodes at any time.

In order to handle large graphs, an alternative to Google Pregel is to use distributed computing. For instance, Jennings & Motyčková [93], in the context of protein structure detection, distribute their algorithm through multiple threads on the same machine. The problem is really tackled with the emergence of Message Passing Interfaces (MPI) [181] implemented on big processing clusters [91, 145]. MPI is a formalisation of an message exchange protocol, that has been implemented on many hardware architectures and programming languages. Almost all the big processing clusters implement the MPI standard. In the MCE fields, Pardalos et al. [145] proposed a distributed implementation of the Carraghan-Pardalos Algorithm [29] with the Message Passing Interface (MPI) standard. Schmidt et al. [174] also use a MPI to distribute the resources among various computers. This algorithm uses a modified version of Koch's MCE, in order to obtain scalability for local (multi-processor and multi-thread) and distributed computation. Schmidt et al. [174] also provided experimental results over graph representation, drawing the conclusion that the adjacency bit matrix representation is the best, in comparison with the linked list or hash map representation. We use this result to represent our graph as bit matrix, when the size of the graph allows it. For large graphs, we prefer a sparse, hash map representation. Nevertheless, their

idea did not tackle the issue of dynamic graphs and data reduction, whereas in some situations the full image of a graph is not needed.

The distributed systems solving the MCE problem have a major importance in the current approaches. The usage of MPI makes possible to process data in a parallel or distributed way. Moreover, by combining these approaches with several heuristics, these systems are able to process larger graphs. Unfortunately, the dynamic issue of the data is not covered by any algorithm and the data reduction hypothesis remains a perspective in the Malewicz et al. [118] article.

Analysis of the Maximal Clique Exploration Algorithms

The choice of an algorithm that deals with large and dynamic data depends on the application. Whereas a static algorithm for clique detection suits well bio-chemestry problems [104], the dynamic approaches are needed for relation detection on social networks for instance, since the structure of a social network can change dramatically [85, 124]. Semantic data graphs [121] are certainly less dynamic, but nevertheless can also benefit from this class of dynamic algorithms.

For small-sized data structures, the existing approaches based on the algorithm proposed by Bron & Kerbosch [24], with the Tomita et al. [196] pivot selection strategy, work well. The authors have not proved that this algorithm can process large data. Moreover, Schmidt et al. [174] show that this algorithm needs to be combined with heuristics in order to be able to process large data structures.

Heuristic algorithms offer an interesting solution to tackle well large graph structures, especially from the processing time perspective. Unfortunately, they do not guarantee the full exploration of the solution space nor the quality of the generated solutions. Thus, heuristic algorithms cannot be used in those research fields where this is a strict condition, for instance when processing linguistic resources.

On the dynamic side of the problem, the work of Stix [185] addresses this problem from a theoretical perspective, without proposing a solution that scale well large data.

As a solution to MCE in large and dynamic graphs, Pardalos et al. [145] and Schmidt et al. [174] propose to distribute the algorithm using an MPI architecture, which decreased the global processing time. Their work does not optimise the memory consumption of the solution representation, nor consider the dynamic character of the data. Malewicz et al. [118] state that graph processing algorithms should be designed in a distributed way, rather than having a single processor. This idea is the foundation of the Google Pregel architecture. Based on this observation, a modern MCE should be parallel and "distributable", to process large data in short time.

Since graphs are a popular representation for instance in bio-chemistry or in semantic data modelling, MCE offers a solution to node clustering and for semantic link exploration. Various research groups tackle the problem using static serial MCE algorithms, whereas others distribute the problem. On recent data structures, large and very dynamic, the state of art approaches seem not to explore well these graphs. A new approach combining all these paradigms needs therefore to be found. In particular,

parallel algorithm seems to be a good solution since high processing time is required for this type of data. So far, none of existing work propose a MCE algorithm that can process large and dynamic data in a parallel way. Furthermore, the memory consumption of the algorithm can be reduced by keeping only the data needed to process a certain exploration case.

We, therefore, propose an algorithm that satisfies all the requirements to process dynamic and large graphs. Moreover, it is a good candidate to process our subtitle corpus data and can be used to generate the contextonyms graph.

3.3 DDMCE Algorithm

3.3.1 Preliminaries

Notations

Some classic notations of graph theory are used hereafter, as described bellow.

- $\mid S \mid$ represents the cardinality of the finite set S.
- A graph is described by $G = (V(G), E(G))$, or $G = (V, E)$ in the short form, where $V(G)$ is the set of vertices/nodes of the graph G and $E(G)$ is the set of edges. Usually, $n = \mid V(G) \mid$.
- For a given $u \in V(G)$, $N(u)$ denotes the set of all the neighbours of u: $N(u) = \{v \mid (u, v) \in E(G)\}$.
- Q denotes the clique set, and q an element of Q.
- A clique is maximal, iff equation 3.1 is satisfied:

$$\forall q \in Q, \nexists q' \in Q, q \subset q' \wedge \mid q \mid < \mid q' \mid \tag{3.1}$$

- A tree $T = (V(T), E(T))$, is considered to be an ordered directed graph, with a root R.
- All the trees have an ordered child list $Ch(p)$, for a given parent p.
- Let $S_L(u) = \{v \in Ch(p) \mid u \in Ch(p) \wedge index(v) < index(u)\}$ be all the siblings that are in the left part of u, and let $S_R(u) = \{v \in Ch(p) \mid u \in Ch(p) \wedge index(v) > index(u)\}$ be all the siblings that are in the right part.
- Let sT_u be the sub-tree associated with the root u, and sG_U as the sub-graph associated to the vertex (node) set U.

The Bron & Kerbosch Algorithm

The DDMCE architecture we propose is based on the Bron & Kerbosch [24] algorithm with pivot selection. Bellow, we present the version of the Bron & Kerbosch algorithm (Algorithm 3.3.1) described by [32].

The initial call of the Algorithm 3.3.1 is done with: explore$(\emptyset, V, \emptyset)$.

Algorithm 3.3.1 Bron-Kerbosch Algorithm with pivot selection

Require: K - partial clique, P - potential node set, D - the explored node set
Ensure: K is a maximal clique, if $P = \emptyset \wedge D = \emptyset$

```
function EXPLORE(K, P, D)
    if P = ∅ ∧ D = ∅ then Report K as a maximal clique
    else
        u_p ← |choosePivot|(P)
        for all v ∈ P \ N(u_p) do
            K ← K ∪ {v}
            P_v ← P ∩ N(v)
            D_v ← D ∩ N(v)
            |explore|(K, P_v, D_v)
            D ← D ∪ {v}
            P ← P \ {v}
            K ← K \ {v}
        end for
    end if
end function
```

The pivot selection has a major role in the exploration strategy since it reduces the exploration space and enables early cuts of the "false" branches on the solution tree, as explained by Koch [103]. Cazals et al. [32] present various strategies to choose the pivot element. According to the literature, and particularly to [32], the strategy proposed by Tomita et al. [196] is the most efficient so far. This strategy (Algorithm 3.3.2) chooses, as long as possible, a pivot that has the maximum number of neighbours inside the potential set of nodes P. If such a choice cannot be made, one of the reasons could be that the nodes contained by P are not connected at all. In the case of a node having the same number of neighbours, a random decision is made.

Algorithm 3.3.2 Tomita et al. Strategy [196] for pivot selection

Require: P - potential node set
Ensure: pivot, according to [196] Strategy

```
function CHOOSEPIVOT(P)
    pivot ← max_{u∈P} | P ∩ N(u) |
end function
```

In an informal description of Algorithm 3.3.2, for each step the pivot is chosen from the potential set of nodes as the node with the highest neighbourhood. Linked with the "for" step from Algorithm 3.3.1 ($v \in P \setminus N(u_p)$), the pivot minimize the exploration set. Tomita et al. [196] gives a formal proof of this observation.

The Bron & Kerbosch [24] algorithm has not been tested on large data, as Tomita et al. [196] present it. Moreover, the algorithm is not adapted for dynamic structure changes, since the algorithm manages a complete image of the graph to process. In this chapter we propose to adapt the original serial Bron & Kerbosch's algorithm [24] with Tomita et al.'s pivot selection strategy [196] into a distributable version.

3.3.2 Algorithm

The Dynamic Distributable Maximal Clique Exploration Algorithm (DDMCE) uses a tree based representation of the solution in order to transform the classic Bron & Kerbosch's algorithm [24] into a distributable version.

The Solution Representation

DDMCE computes a tree-based decomposition from the initial graph representation to encode the solution of the MCE problem. As presented in Figure 3.1, the algorithm starts with an artificially introduced root named $\{*\}$ and on which the exploration algorithm is applied, that generates the tree. As stated before, one central component of the algorithm is the pivot, a node which has the property of balancing the solution tree. From the pivot selection perspective, the Tomita et al.'s strategy [196], listed as Algorithm 3.3.2, is proven to be the most efficient [32]. On Figure 3.1, the pivots selected thanks to the Tomita et al.'s strategy are highlighted in boxes placed at the right of the tree nodes.

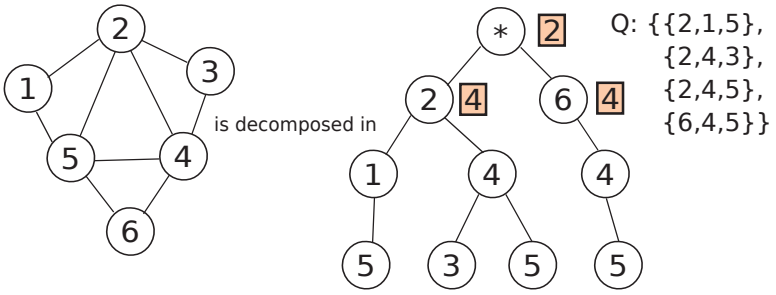


Figure 3.1: Graph to tree transformation, done with DDMCE, using the Tomita et al.'s pivot selection [196]. The cliques found are presented in the Q set and the pivot at each step is presented in the highlighted boxes.

The tree-based structure ensures that each path from root to leaves is optimal. With such a representation, all suboptimal solutions are eliminated at construction time. A new node is added in the solution tree at each iteration of the algorithm and when a node is invalidated, it is marked for removal. On the other hand, when a node is known to be part of a final (optimal) solution, the node and the whole path is marked as final. Thus, the optimality of the tree is guaranteed at any time.

DDMCE Algorithm

A complete listing of the DDMCE algorithm is available as Algorithm 3.3.3.

This algorithm is initialized with a potential node set, the explored node set and the current vertex which needs to be expanded. The potential node set and explored set are strictly linked with the currently processed vertex. This relation enables parallelization, as two nodes can be processed independently, since their potential set and explored set are independent.

In the case of an empty potential set, which corresponds to the final steps of the algorithm, the current node could be final or the node need to be removed because the

Algorithm 3.3.3 DDMCE algorithm

Require: P - potential node set, D - explored node set, vi - current vertex from tree representation
Ensure: vi is marked as final, if $P = \emptyset \wedge D = \emptyset$

```
function EXPLORE(P, D, vi)
    if P = ∅ then
        if D = ∅ then
            |markFinal|(vi)
            |propagateFinal|(vi)
        else
            |markRemoval|(vi)
        end if
    else
        u_p ← |choosePivot|(P)
        |markPivot|(vi, u_p)
        for all v ∈ P \ N(u_p) do
            ve ← |createNode|(v)
            P_v ← P ∩ N(v)
            D_v ← D ∩ N(v)
            |parallelExplore|(P_v, D_v, ve)
            D ← D ∪ {v}
            P ← P \ {v}
        end for
    end if
end function
```

solution is suboptimal. Optimal solutions are obtained when the already explored and the potential set are empty ($P = \emptyset \wedge D = \emptyset$). In the case of suboptimal solutions, the current node is marked for removal.

The functions *markFinal* and *propagateFinal* mark the current node from the solution tree as final, and the node becomes part of an optimal solution. Thereafter, the final mark is propagated to the whole path, up to the root, labelling the whole clique as optimal. In the case of sub-optimality, the current node is removed with the *markRemoval* function.

Afterwards, the *parallelExplore* function wraps the whole data on a new parallel call, which is either transmitted to a new processor or processed locally, according to the number of available processors used for the task. In the end, for each parallel call, the same explore function is used.

For instance, when exploring the graph described in Figure 3.1, the Algorithm 3.3.3 produces the following partial results:

Step 1: $vi = *$, $P = \{1, 2, 3, 4, 5, 6\}$, $D = \emptyset$, $pivot = 2$

Step 2: $vi = 2$, $P = \{1, 3, 4, 5\}$, $D = \emptyset$, $pivot = 4$

Step 3: $vi = 1$, $P = \{5\}$, $D = \emptyset$, $pivot = null$

Step 4: $vi = 5$, $P = \emptyset$, $D = \emptyset \rightsquigarrow vi = 5$ - marked as final $\rightsquigarrow \{5, 1, 2\}$ - marked as final

Step 5: $vi = 4$, $P = \{3, 5\}$, $D = \{1\}$, $pivot = null$

...

At the end of the exploration, the full tree is marked as final and the $\{2, 1, 5\}$, $\{2, 4, 3\}$, $\{2, 4, 5\}$, $\{6, 4, 5\}$ sets are also marked as final. These sets correspond to the clique solutions found by our algorithm.

Before presenting the implementation of our system, we would like to introduce a series of set definitions, concerning the solution (previously introduced as K in Algorithm 3.3.1), the potential exploration (P) and the already explored (D) set. This is followed by a series of observations on the dynamic set transformations and a brief discussion over the formalisation part.

Set definitions

Due to the tree representation of the solution, the three K, P and D sets used in Bron & Kerbosch Algorithm, can be computed at runtime, reducing the storage necessities.

- K set, representing the current detected clique, is contained in the tree representation: as every path in the solution tree represents an optimal clique detected, a given K set could be found as a path in the tree.

- P set, which contains all the potential nodes, can be computed with the following formula:

$$P_u = \begin{cases} V & \text{if u = R (R is the root node of the solution tree)} \\ P_p \cap N(u) \setminus S_L(u) & \text{with p the parent of the node u} \end{cases} \tag{3.2}$$

Proof. The first statement $P_R = V$ occurs at the algorithm initialisation for $P \leftarrow V$, when $vi \leftarrow R$. The more generic P_u definition occurs at the construction of the new child nodes (v):

$$P_u = P_{last} \cap N(u) \tag{3.3}$$

$$P_{last} = P_p \setminus S_{processed} \tag{3.4}$$

$$S_{processed} = \{v \in P_p \setminus N(pivot) | v - \text{has been previously processed} \wedge p - \text{is the parent of } v\} \tag{3.5}$$

According to tree set definitions and tree construction, $S_{processed}$ is equivalent with $S_L(u)$. Following equations 3.3,3.4 and 3.5 the P_u equation can be rewritten as: $P_u = P_p \cap N(u) \setminus S_L(u)$, for $u \neq R$. □

- D_u set, representing the nodes already processed, can also be computed at runtime:

$$D_u = \begin{cases} \emptyset & \text{if u = R (R is the root node of the solution tree)} \\ D_p \cap N(u) \cup S_L(u) & \text{with p the parent of the node u} \end{cases} \tag{3.6}$$

Proof. Similarly to previous proof, the statement $D_R = \emptyset$, corresponds to the algorithm initialisation with $D \leftarrow \emptyset$, when $vi \leftarrow R$. The D_u definition is obtained as follows:

$$D_u = D_{last} \cap N(u) \tag{3.7}$$

$$D_{last} = D_p \cup S_{processed} \tag{3.8}$$

$$S_{processed} = \{v \in P_p \setminus N(pivot) | v - \text{has been previously processed} \wedge p - \text{is the parent of } v\} \tag{3.9}$$

According to tree set definitions and tree construction, $S_{processed}$ is equivalent with $S_L(u)$. Following equations 3.7,3.8 and 3.9 the D_u equation can be rewritten as: $D_u = D_p \cap N(u) \cup S_L(u)$, for $u \neq R$. □

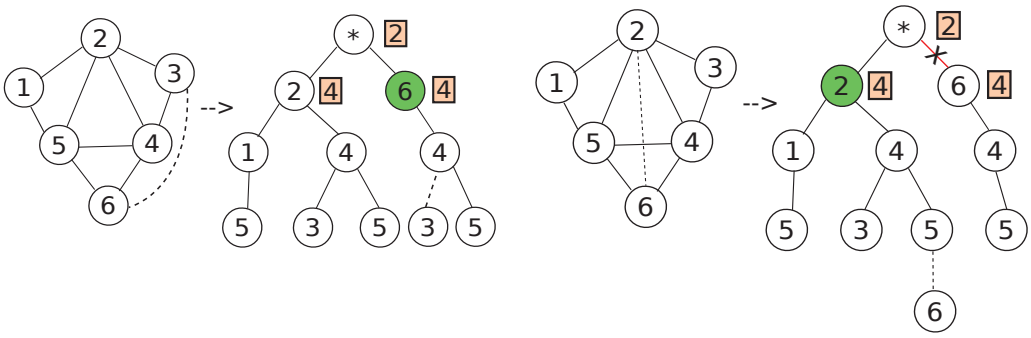
Dynamic transformation

Thanks to the tree based representation, the whole graph does not need to be reprocessed in case of modification. The dynamic operations considered are edge addition and edge deletion, since node operations are generalisation of the corresponding edge operators. Dynamic changes in the graph structure can lead to partial inconsistency in the solution tree. Therefore, all the branches in the tree that have been affected by these changes have to be recomputed and revalidated.

Figure 3.2 shows that when the graph changes, the nodes of the tree mutate according to the new representation. In Figure 3.2a, by adding a new edge between nodes 3 and 6, where neither of them were selected as a pivot, only the sub-tree having 6 as root needs to be recomputed. The vertex 3 is currently only a leaf in this decomposition, so it does not need to be revalidated. In Figure 3.2b, the edge is added between nodes 2 and 6, and 2 was selected pivot for the root, the sub-tree given by node 6 needs to be removed and the one given by node 2 needs to be revalidated. This case represents also a worst-complexity example, because the revalidation decision taken is equivalent with the root revalidation. This scenario may happen in the case of very dense graphs, or simply with near-complete nodes.

More formally, for a new edge (e=(n_1, n_2)), two different situations can be observed:

- In the case of an addition of a regular edge (without pivot influence, Figure 3.2a), all the sub-trees having $n_1 \vee n_2$ as roots have to be reprocessed.
- When one of the nodes involved in the edge operation is a pivot in the graph (Figure 3.2b), all the branches from the other node are removed and the sub-tree



(a) Graph Transformation by adding a new edge

(b) Graph Transformation by adding a new edge (pivot influence)

Figure 3.2: The graph to tree of the clique algorithm during dynamic transformation. The highlighted (green) nodes need to be reactivated in order to recompute the tree.

starting from the pivot is recomputed. In other words, each new neighbour of the pivot becomes part of its sub-tree.

Discussion

Several observations can be made:

- The P set decreases on every call of the *explore* function (see Algorithm 3.3.3). For instance, a processing unit which is exploring node $\{6\}$, with the associated solution tree sT_6, needs to explore only the sub-graph given by $sG_{P_6 \cup D_6}$. In other words, while a part of the solution is processed, the full graph representation is not required. The sub-graph given by the $P_6 \cup D_6$ vertices is sufficient.
- The chosen pivots need to be remembered on the solution tree, since they have a strong influence on the tree representation.
- The *explore* function is designed so that it does not need to run on the same machine or thread, according to the distributed implementation.
- The tree-based structure of the solution allows a distributed processing strategy due to the data independence of all the children. After the processing of the parent node (e.q. root node), all the children can be processed at the same time (distributed or parallel processing).

A good implementation of a distributed algorithm has to take into account all these heuristics, since they would boost the performance of the system.

3.3.3 Implementation

Architecture

The DDMCE algorithm is implemented within a Message Processing Interface (MPI) architecture proposed by [181] (see Figure 3.3). The message is designed to be fast and contains sufficient information to expand the solution tree.

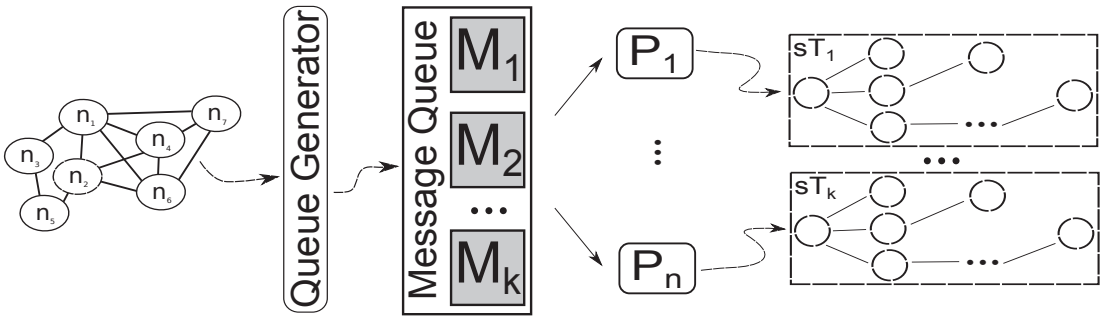


Figure 3.3: The distributed processing pipeline.The master process generates the message queue $\{M_1...M_k\}$. P_i represents the i-th processor which takes a message and generates the solution sub-tree attached to the corresponding M_j. In this setup we have k messages and n processors, with $k \gg n$.

First of all, the Queue Generator receives the graph structure and decomposes it: a sub-tree to process by message. Then, the resulting message queue is stored as a shared data in the memory. Each sub-tree is processed by a processor (also called worker). In this example, k messages are generated and processed by P_n workers (n fixed by the hardware configuration used). Each P_i generates a solution to its own sub-problem, starting from the M_j given message and having as output a sT_j solution sub-tree. Finally, all the sub-trees are merged in order to reconstitute the solution.

To avoid node synchronization issues, the graph data is read-only for the workers, so that data cannot be modified by the algorithm in any way. The only graph modifications allowed during the process are external from the algorithm and corresponds to the dynamic character of the graph.

In our design, two types of workers are used: a worker having the specialised task of generating the message queue, starting from the graph, and workers using multiple processors to generate the final solution. Fundamentally, the same Algorithm 3.3.3 is executed by both of the workers, only the input data type changes. At the initialization step, when the message queue is created, the P set contains all the nodes of the graph. While the Queue Generator is running, each call to *parallelExplore* is pushing a new message to the queue. At runtime, the Queue Generator remains latent in order to process any external change in the graph structures and to push new messages according to graph modifications. Whereas the Queue Generator is in charge of generating messages, the other workers apply the Algorithm 3.3.3 and generate the solution tree locally. Every call to *parallelExplore* creates a local explore message. After all the solution sub-trees are computed, the merge is easily done, since all sub-trees are inserted at the root node, called {*} in the Figure 3.1.

One of the major components involved in the data exchange mechanism and processing steps is the message architecture. In general, in order to gain speed, a message should be designed to encapsulate all the useful information. Nevertheless, redundant information should be avoided in order to increase the transport speed. In our approach, the message contains a tree node, which encapsulates the two sets used in the algorithm: P (potential set) and D (already explored). Moreover, it also contains the current node (u) and the pivot (p).

Strong points

The tree representation is robust to structure changes, even if these changes occur in real time. All the structural modifications are computed at runtime by the Queue Generator, and are injected into the message queue. In other words, a modification to a node or to an edge in the graph is modelled as a sub-tree revalidation in the solution tree, and leads to add the corresponding node to the sub-tree in the processing queue.

Similarly to dynamic situations, DDMCE offers a high re-usability of a given solution for a graph. In case of graph modifications after a complete process by DDMCE, only partial computation, linked with the inconsistent or partial sub parts of the solution, is needed. In more dramatic scenarios, such as hardware failure, lost nodes could be recomputed as well. In fact, in order to recalculate partial cliques with DDMCE, the only verification that should be done is the lookup of inconsistent node, which usually takes less time than applying the whole algorithm from scratch. A partial solution is obtained when the exploration tree is incomplete. Finding inconsistent nodes requires only the exploration of the last level of the solution sub-tree to detect if the P set associated to each node is not empty.

Based on the space reduction observation (first paragraph in Section 3.3.2), our algorithm does not need a full image of the graph for every processor. While exploring a new solution, having a node with an exploration set (P) and the nodes already explored (D set) associated, only the image of the sub-graph attached to these two sets are stored in memory.

Finally, the graph exploration problem is decomposed into a Depth-First tree exploration by DDMCE, which is more efficient than the Breadth-First strategy. As a clique is represented by a path from the root to a leaf in the tree representation, a depth-first exploration enables to remove every cliques found from the tree, once processed, and therefore to save memory.

Technical details

The DDMCE algorithm is implemented using Java programming language (Oracle distribution, version 1.7.0 for Linux x64 platforms) and Colt library [33] (version 1.2.0).

The implementation results rely on a simulated version of a distributed environment, where the whole code runs on a multi-core machine. Even if the implementation can be considered rather parallel than distributed, for the moment, the design has been made having in mind the distributable aspect of the algorithm.

To obtain a fully distributed algorithm, the only structure that should be modified is the Message Queue. Currently, two different implementations are proposed, the first one is based on a shared queue maintained entirely in memory, that suits well small graphs (less than 1 million cliques), and the second one is based on a shared MySQL database ([206]), that is used to process larger structures.

This architecture is evaluated over several experimental setups, covering from simple static experiments, to parallel version and dynamic cases.

3.3.4 Experiments

In order to evaluate DDMCE, experiments have been conducted on several graphs. DIMACS Database ([96]) contains graph examples from various fields, some obtained from real data (i.e. Hamming graphs) and others obtained by using strategy-based generators (i.e. Johnson graphs). Unfortunately, DIMACS does not suit well our purpose since the proposed graphs are very small and most of the algorithms designed for this database have been finely tuned to solve such a cases. We therefore decided to test the DDMCE algorithm against the best state of art algorithm, proposed by Tomita et al. [196], on large and randomly generated dynamic graphs.

The random graph generator used is a simple edge builder based on linear node sampling in order to create a neighbourhood. It receives as input the number of desired nodes and a density, and it builds an uniform distributed graph with variance for density of around 1%. The same algorithm is also used to generate extra nodes when simulating the dynamic effect.

The experimental setup have been tested on a Dell I7 machine (8 cores, with 1.6 Ghz per core), with Ubuntu Linux, x64 version. A second Xeon machine has also been used to test the dynamic part of the experiments, having 4 cores at 1.8 Ghz per core. The Java Virtual Machine (version 1.7.x for Linux x64) offered by Oracle enabled to obtain the best speed offered by a Java platform in both configurations.

Multiple graphs have been built, with the same number of nodes and densities, to evaluate DDMCE performances. For all these graphs, the mean ($\bar{x}$) and variance (σ^2) are computed for both the running time and number of cliques. In order to compare the DDMCE results with the CLIQUES algorithm [196] results, the same measure, called $/clique$, is also used during the experiments. It describes the average time needed to process 10^6 cliques. This measure is computed at each run. More formally, the measure can be calculated using the equation 3.10, where $time$ represents the running time of the algorithm (in seconds) and $\#cliques$ is the number of cliques.

$$/clique = \frac{time \times 10^6}{\#cliques} \tag{3.10}$$

Table 3.1 presents a comparison between the mean of DDMCE running times and the ones presented by Tomita et al. [196] with CLIQUES Algorithm. It should be noted that, for a robust estimation, the running times presented for DDMCE are the average of ten runs on different graph configurations (same density and number of nodes, but different number of cliques), whereas the running times of the CLIQUES Algorithm correspond to a single configuration. For most of the density configurations, our algorithm outperforms CLIQUES (marked in bold). For the other densities, the running times are approximately the same.

The second experiment concerns a dynamic environment that has been simulated by adding a new node to the graph during the processing. Each newly added node has the same neighbourhood density, as for the static case. The running times are presented

Generated		DDMCE			CLIQUES [196]		
n	ρ	#cliques	time(s)	/clique	#cliques	time(s)	/clique
1,000	0.01	4,723	0.058	12.36			
	0.03	10,486	0.122	11.65			
	0.05	21,158	0.175	8.26			
	0.07	45,846	0.280	6.11			
	0.10	99,672	0.503	5.04	99,062	0.210	**2.12**
	0.15	349,313	1.086	3.11			
	0.20	1,203,454	3.369	2.80	1,183,584	2.250	**1.90**
	0.25	4,299,261	12.008	2.79			
	0.30	15,935,818	57.386	3.60	15,362,096	33.180	**2.16**
5,000	0.01	96,514	0.617	**6.40**	49,738	10.860	218.34
	0.03	514,312	2.045	**3.98**	141,651	11.180	78.93
	0.05	1,789,995	6.988	**3.90**	215,129	11.740	54.57
	0.07	4,081,340	18.635	**4.57**			
	0.10	18,442,189	95.025	5.15	18,483,855	86.600	**4.69**
10,000	0.01	347,898	3.003	**8.63**	348,552	14.780	42.40
	0.03	3,741,993	19.004	**5.08**	3,738,814	41.290	11.04
	0.05	12,182,435	98.022	**8.05**	12,139,182	109.780	9.04
	0.07	42,705,673	398.129	9.32	42,572,404	338.230	**7.94**
	0.10	228,848,693	2,527.744	11.05	229,786,397	1,825.450	**7.94**

Table 3.1: Running times for random generated graphs. n is the number of nodes and ρ is the density. The values presented for DDMCE are the average of the #cliques and time(s) measures, whereas for CLIQUES algorithm the values are those presented in Tomita et al. article [196].

in Table 3.2. The $T_{dynamic}$ and T_{static} times are defined in seconds and the $/dynamic$ measure is given by the equation 3.11:

$$/dynamic = \frac{T_{dynamic}}{T_{static}} \tag{3.11}$$

In a dynamic context, the maximal running time is equal to the time needed to compute all the cliques in a static context. This happens when a node with a very dense neighbourhood is dynamically added, which triggers an update on the whole tree representation. Fortunately, on real data (for instance social networks), such nodes are very rare [124].

A clear image over the computation times is needed, therefore the experiments are conducted on a multi-core platform. The serial case is carried out using a single core exploration, whereas the parallel case is managed with multiple cores (2 or 4). During

n	ρ	c	T_{Static}	$T_{Dynamic}$	/dynamic
1,000	.01	1	.342	.029	.084
	.01	2	.250	.021	.086
	.01	4	.240	.022	.092
	.03	1	.598	.061	.102
	.03	2	.557	.043	.077
	.03	4	.543	.043	.078
	.05	1	.776	.119	.154
	.05	2	.836	.098	.117
	.05	4	.996	.101	.101
	.07	1	.992	.210	.212
	.07	2	.886	.164	.185
	.07	4	1.071	.160	.149
	.10	1	1.617	.483	.298
	.10	2	1.215	.388	.319
	.10	4	1.433	.365	.255
	.15	1	4.683	1.037	.221
	.15	2	3.217	.906	.281
	.15	4	3.266	1.015	.311
	.20	1	16.819	2.360	.140
	.20	2	9.527	1.852	.194
	.20	4	9.781	2.069	.211
	.25	1	67.404	8.501	.126
	.25	2	37.192	5.381	.145
	.25	4	35.882	5.665	.158
	.30	1	281.932	40.488	.144
	.30	2	160.485	22.151	.138
	.30	4	149.339	21.711	.145

n	ρ	c	T_{Static}	$T_{Dynamic}$	/dynamic
5,000	.01	1	1.687	.125	.074
	.01	2	1.266	.099	.078
	.01	4	1.324	.100	.076
	.03	1	9.010	.541	.060
	.03	2	5.673	.449	.079
	.03	4	5.363	.432	.081
	.05	1	30.805	1.267	.041
	.05	2	18.585	1.103	.059
	.05	4	17.828	1.102	.062
	.07	1	90.080	3.854	.043
	.07	2	54.499	2.804	.051
	.07	4	52.718	2.746	.052
	.10	1	408.919	19.324	.047
	.10	2	245.741	11.875	.048
	.10	4	235.444	11.580	.049
10,000	.01	1	13.114	.292	.022
	.01	2	9.522	.228	.024
	.01	4	9.246	.194	.021
	.03	1	99.625	1.788	.018
	.03	2	62.055	1.405	.023
	.03	4	57.786	1.593	.028
	.05	1	467.644	11.138	.024
	.05	2	286.653	7.247	.025
	.05	4	269.613	6.687	.025
	.07	1	1,768.328	59.519	.034
	.07	2	1,075.273	37.237	.035
	.07	4	1,015.318	34.676	.034
	.10	1	10,550.762	491.325	.047
	.10	2	6,379.845	301.977	.047
	.10	4	6,068.725	282.626	.047

Table 3.2: Running times for static and dynamic graphs. n represents the number of nodes, ρ is the density and c the number of cores. All the times ($T_{dynamic}$ and T_{static}) presented are in seconds. The */dynamic* measure is defined by the equation 3.11.

the dynamic experiment, the exploration was done on a single core. Table 3.2 and Figure 3.4 summarise all the results of this experiment. In order to evaluate the time differences between static and dynamic computation, the ratio between the two times is computed (equation 3.11). This measure is on average 17 % ($\sigma^2 = 0.07$) for small graphs (n=1,000 nodes), and 4.7 % ($\sigma^2 = 0.02$) for larger ones, as it can be deduced from Table 3.2.

Figure 3.4 synthesizes the comparison of running times for different densities (ρ) and graph sizes (n). The time is represented on an logarithmic scale in order to cover all the running intervals for all the densities.

The computing times obtained for the parallel scenario is lower, in comparison to the same set-up running on serial architectures. In dynamic contexts, the times decrease more, showing that our approach is efficient to process dynamic graphs. Figure 3.4 illustrates the dynamic difference, which in case of large graph reaches 4.7%, and 17% for small graphs. In conclusion, DDMCE produces results up to 20 times faster in dynamic context than recomputing the whole solution, in a single thread architecture. The complete results obtained by DDMCE are summarized in Appendix B and C.

Based on these results, this algorithm is a good candidate to process the context graph that would generate the affective contextonyms resource. The word co-

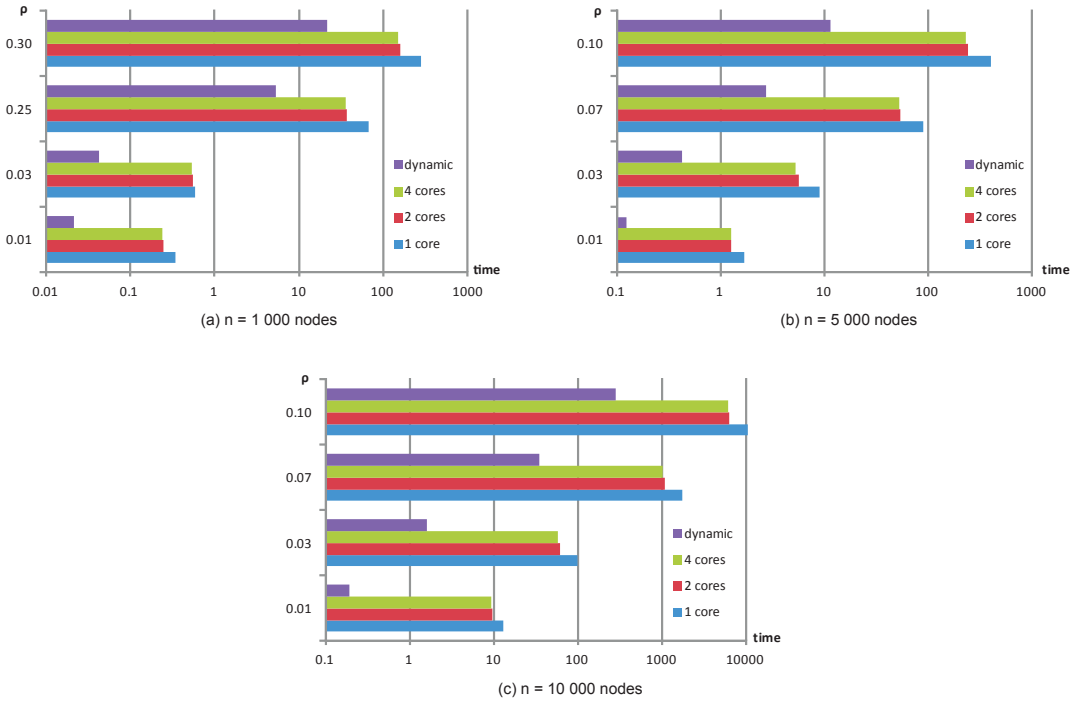


Figure 3.4: Running time comparison for several graph size samples (n) and for different densities (ρ), given on an logarithmic scale representing time.

occurrences generate a very large structure. Moreover, since more documents can be added at any time, the dynamic co-occurrence graph can be easily regenerated by DDMCE at any point.

3.4 A new linguistic resource: affective contextonyms

Sentiment analysis and affect detection algorithms are generally based on annotated data, structured into dictionaries, ontologies or word nets. Among other research problems, two issues are considered very important in this field: 1) word sense disambiguation and 2) accuracy of the affect detection.

Most of the current approaches use annotated resources based on word nets. Their structure, founded on synonymic relations, makes the disambiguation process very difficult. Our model uses contextonyms, which simplify the decision process. Therefore, the disambiguation issue is transformed into a context matching problem. The second focus is on the manual annotation of the data followed by a semantic valence propagation. This approach enables to obtain, through the expansion process, new affective labels from a set of initial ones. Unfortunately, this is usually done to the detriment of the precision.

3.4.1 SentiWordNet

Among other WN extensions, SWN [11] has been built automatically by using a valence propagation technique over WN. It has been designed as a lexical resource for valence

prediction of a sentence, for applications in opinion mining and sentiment analysis. SWN contains annotations for mainly all the WN 3.0 synsets, introducing for each of them a degree of positivity, negativity or objectivity. Each of these valences are defined on a scale from 0.00 to 1.00, with the sum of all three of them being 1.00.

Figure 3.5 presents the synset (id=a#00064787) associated to the word **"good"**, annotated according to SWN. This word belongs to 27 synsets in WN and SWN (21 as a adjective, 4 as a noun and 2 as an adverb). For the selected synset, **"good"** has a positive value of 0.625 and an objective valence of 0.375. The authors of SWN propose a triangle based visualisation, where each corner represents a different sub-value of the valence: Positive (P), Negative (N) and Objective (O).

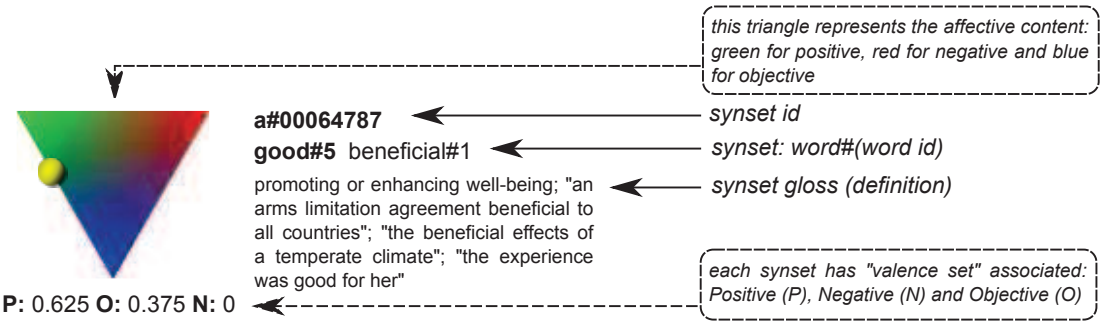


Figure 3.5: A SWN [11] example, also containing the visualisation model. P states the positive degree, N the negative and O the objective one.

Semantic valence propagation

The method of semantic valence propagation refers to diffusion of a valence through a structured (graph or synset relations) corpus. The spreading is done by respecting the links between the words. The structures tested so far [11, 161, 66] are different versions of WN, using the synsets. In very frequent scenarios, the process starts with a manually annotated set of words (also called "seeds") and on each iteration the valence of these seeds is spread on the network. C. Potts gives a more formalised definition of the algorithm in the Sentiment Analysis Tutorial [156].

A more complex approach involves weighted propagation of valences [19]. The weights are usually the word frequencies inside the corpus. The spreading of the valences is done in the neighbourhood of the seed nodes according to these weights. This measure can be extended further by taking into account the neighbourhood density and node frequency.

SWN is the largest dictionary generated using valence propagation. For instance, compared to the largest manually annotated dictionary, Harvard General Inquirer [186], SWN has almost 10 times more annotated words (see Table 3.3), which leads to a better coverage over the affective vocabulary. However, there is a high disagreement between SWN and other linguistic resources. It contains conflicts and inconsistencies inherited from the WN structure that cannot represent multiple contexts of use [156].

SentiWordNet disagreements with other manual resources

C. Potts [156] conducted a study of SWN disagreements with several other opinion mining resources (see Table 3.3). Since all the other dictionaries offer discrete labels for valence, Potts measured the overlap between the maximal valence extracted from SWN and the labels of other dictionaries.

Dictionary	Disagreement	Annotation Style	Word Count
MPQA [207]	27%	+/-	8,221
Opinion Lexicon [112]	25%	+/-	6,789
General Inquirer [186]	23%	+/-	11,788
LIWC [190]	25%	Psychometric Categ.	4,500
SWN [11]	-	Continuous Valence	**117,659**

Table 3.3: Disagreement level, according to C. Potts [156], between SWN and several other corpora

MPQA (Multi-Perspective Question Answering) Subjectivity Lexicon [207] contains annotations based on the subjectivity level, part of speech and polarity. Polarity corresponds to a discrete valence annotation, having a different label for positive or negative. Opinion Lexicon is maintained by Bing Liu [112] and contains discrete manual annotations for positive and negative words. Harvard General Inquirer [186] is a lexical resource which is concentrated in attaching syntactic, semantic and pragmatic information to part-of-speech tagged words. It contains positive, negative and hostile[1] labels for most of its containing words. Finally, Linguistic Inquiry and Word Counts (LIWC) [190] is a proprietary database, containing categorised words to their psycho-semantic state, which can be translated into negative or positive labels.

SWN presents an average of 25 % disagreement with MPQA, Opinion Lexicon, Harvard General Inquirer or LIWC. These disagreements between SWN and other corpora are due to the construction of SWN, based on automatic semantic propagation.

SentiWordNet ambiguities and inconsistencies

SWN contains conflictual valences for the same word, that corresponds to two distinct situations:

1. a word has different valences among different synsets (inter-synset conflict),

2. a word has conflictual valences within the same synset (intra-synset inconsistency).

For instance, the '*heart*' synsets, extracted from SWN, contains inter- and intra-valence inconsistencies:

[1] According to Harvard General Inquirer [186], a subset of 833 words are tagged *Hostile*. There words are indicating an attitude or concern with hostility or aggressiveness. In our approach, we consider these labels as a sub-set of the negative ones.

1. spirit#8 **heart#6**: *an inclination or tendency of a certain kind; "he had a change of heart"*, **+0.5**

2. **heart#1** bosom#5: *the locus of feelings and intuitions; "in your heart you know it is true"*, **-0.125**

3. spunk#2 nerve#2 mettle#1 **heart#3**: *the courage to carry on; "you haven't got the heart for baseball"*, **+0.25 -0.25**

Synsets 1) and 2) show an inter-synset inconsistency, since the valence of **heart** in 1) is positive, whereas in 2) is negative. In example 3), the inconsistency exists within the same synset.

The first issue can be solved using context. Considering that each synset corresponds to a particular meaning, then the valence from SWN is applied to the chosen context. In practice, finding the proper context only with WN synsets is quite challenging.

On the contrary, the second type of conflict is an artefact of semantic propagation algorithm of SWN. A word, within a synset, should not have conflictual valences because it would lead to ambiguous decisions. In practice, this problem is similar to the first case, because a term with conflictual valences would have two different contexts.

A short statistical analysis highlights that 10,939 words (out of 117,659) from SWN carry conflictual valences, among which 9,643 words are having conflictual valences in the same synset. These conflicts represent 9.29% of the whole corpus. Part of these conflicts are linked with the disagreement levels according to Potts [156].

SentiWordNet and context

WN is one of the widely used linguistic resources in natural language processing applications. Even so, grouping words into synonymic relations makes very difficult the decision of choosing the right meaning of a term for a given context. On the other hand, SWN has been built using automatic semantic propagation over WN. This lead to the construction of the largest linguistic resource for Sentiment Analysis, with the drawback of having multiple valences associated to the same synset.

Other linguistic resources, which have been manually annotated, have fixed the context of each word to the most common one. This lead to very low disagreement between these resources.

In our model, we introduce the context on each SWN word, which decreases the inconsistency of the dictionary. This is done in a similar way to the manual annotation, by fixing the context, but rather than doing it manually (which is a time consuming task), we are building it automatically.

3.4.2 Modelling context with contextonyms

Contextonyms were introduced by Ji et al. [94] to model contextual use of words. A contextonym graph links words according to word co-occurrences in a certain window[2]

[2]Usually, the window size is fixed to 5 words [94].

and is represented by a network with words as nodes and co-occurring frequencies as edges. In order to extract strong relations between words, a clique exploration algorithm can be applied to a contextonym graph. The collected cliques, that correspond to strong context of use, are called "contextonyms" [94]. As a reminder, a clique is represented by a complete sub-graph or, in other words, a sub-graph in which each node is connected to all the other nodes of the clique. In a contextonym graph, a clique summarizes the strong semantic links between the words that composed it, and therefore could be considered as a context of use [94]. Contrary to synsets, all the words of a contextonym are not equivalent since they are weighted by pairs according to their co-occurrences.

A contextonym model uses a textual corpus as support. The extracted contexts are therefore representative of the corpus. From this textual corpus, a word co-occurrence graph is constructed: the contextonym graph. To construct a contextonym model, we propose a four-step process.

Preprocessing step

The first step, called preprocessing, consists in filtering the text information of the chosen corpus, in order to remove any useless information such as special characters, punctuation, camel-case separators and stop words. Are considered as stop words all the prepositions, articles and other short words[3] that do not carry any contextual semantic value.

Contextonym graph extraction

A co-occurrence network is then constructed, that corresponds to the contextonym graph, counting the word co-occurrences in a certain window within the filtered textual corpus. These occurrences are extracted for each pivot word, by fixing a window size of 5 words. In other words, the co-occurrences are counted for the two words preceding and two worlds following the pivot.

Node filtering

In order to reduce the noise, several filtering techniques are proposed by Ji et al. [94]: 1) a global filter, which eliminates all the nodes that occur very rarely in the corpus, 2) a local filter, which is applied to every node and remove the neighbours with a low occurrence and 3) a child filter, which is similar to the local filter but it is applied to the neighbours of every nodes. In our approach, we applied the global filtering technique, by removing very low word frequencies from the graph. The other two filters are used to delete very low word co-occurrences from the model.

[3]The stop word collection we use to build a contextonym model is available at `http://www.textfixer.com/resources/common-english-words.txt`.

Clique extraction

Finally, a clique exploration algorithm is applied to the contextonym graph in order to extract the contextonym model. We use the DDMCE algorithm, previously presented, dedicated to clique exploration on large and dynamic data. These approach is applied on our corpus and all the collected cliques are included the contextonym model.

3.4.3 A new linguistic resource: an affective contextonym model for dialogues

An accurate linguistic model is important for most of the applications based on natural language processing. Therefore, the choice of the corpus used to compile our model is critical.

The first option was Project Gutenberg[4], due to the large amount of free e-books available at this source. Moreover, these documents are very trustworthy since most of the books are constantly reviewed by the community for spelling or formatting errors. Unfortunately, the vocabulary used within this corpus is too formal for a dialogue context.

In order to focus on modern spoken language, we compiled a large movie subtitle corpus from multiple sources: the Open Subtitle and Podnapisi Archives[5]. The quality of the files was assumed from the total number of downloads and the author's rank on the website. Moreover, old subtitles (year < 1990) were filtered in order to ensure a modern (up-to-date) vocabulary. Finally, a total of 53,384 movie subtitle files are kept for the corpus.

Contextonyms

During the preprocessing step, filters specific to sentence tokenizing on subtitle files were also applied to remove all the time synchronisation data, as well as advertisements. Even if the SubRip[6] format is clean and simple, a template validation has been performed to ensure the integrity of the data extracted. From a space reduction perspective, only the words carrying a strong semantic and emotional value (e.g. nouns, verbs, adverbs and adjectives) are kept, as WNA is suggesting [188]. This filter can be considered as a key word extractor.

After the node filtering step, 86,276 words (words whose frequency is higher than 0.01%, which corresponds to the global filtering technique) and 3,948,359 co-occurrences (with a frequency higher than 0.01 %, done by applying the local and child filter) compose the contextonym graph. On this data, the DDMCE algorithm was applied, extracting a total of 702,546 contextonyms (cliques) that compose our model.

[4]Available at: `http://www.gutenberg.org/`

[5]The corpus represents a part of the subtitle database from `http://www.opensubtitles.org/` and `http://www.podnapisi.net/`

[6]SubRip (.srt) is a very basic text format used to encode subtitle files.

A word modelled as an affective contextonym

Once the contextonym model is built, the valences from SWN could be added to each word. Even if the valences are attached to synsets in SWN, we do not intend to map the synsets to our contextonyms model, because this would be a very difficult task in the context of large context graphs. Our purpose is to preserve the existing valences attached to each word (as part of various synsets) and attach them to our context model, without modifying the actual valence, if possible.

We consider that each contextonym could not have conflictual valences (multiple values for the same word or opposite valences inside the same contextonym). In the case of a conflict, these are solved by choosing a single value for each conflictual word.

For instance, the word "heart" from the previous example has a high frequency in SWN. In Figure 3.6, we present all the contextonyms associated to this word. They have been extracted from our subtitle corpus, while the labels are given by SWN. The word has a neighbourhood of 5126 words and is part of 52 cliques, with a size varying from 4 to 7 words. Moreover, the same word has 418 cliques associated with a size from 2 to 7. Originally, the clique size of 2 and 3 where filtered, but later they proved very useful in practical situations.

One example of such a contextonym is given by: *sadness* (valence = -0.75), *emptiness* (valence = -0.75), **heart** (conflictual valence), *love* (valence = +0.39). This is considered conflictual since the word "heart" is ambiguous. A second inconsistency is given by the presence of two negative words (sadness, emptiness) and one positive (love).

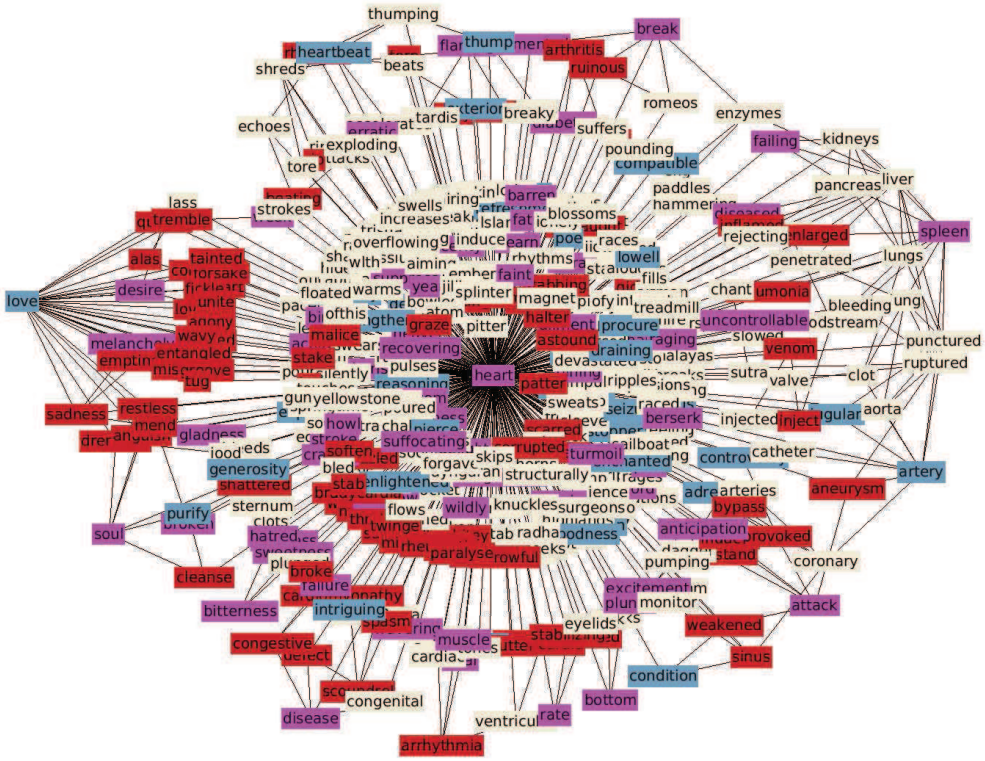


Figure 3.6: A fully annotated contextonym graph, representing the whole neighbourhood of the word '*heart*'. The labels are coloured according to their valence: blue for positive, red for negative, purple for mixed-value (conflictual valences) and light-grey for neutral.

Conflict solving algorithm

Our contextonyms model contains 702,546 cliques, from which 354,109 (50 %) are considered conflictual.

A clique ($\mathcal{Q}$) is considered conflictual if more than one opposite valence can be chosen for each contained word (w). In order to establish a common measure for every possible choice of a valence on a certain node, a dominant has to be defined, as given by the equation 3.12:

$$dominant = \max_{s \in \mathcal{S}} \sum_{w \in \mathcal{Q}_s} freq(w) \times valence^*(w) \tag{3.12}$$

where we define the set $\mathcal{S}$ as containing all the possible valences of a word: positive, negative or neutral. $\mathcal{Q}_s$ represents the sub-clique containing words of a given valence s. The $valence^*(w)$ represents the valence chosen for the word w at the moment of the computation. The $freq(w)$ is the relative frequency associated to each word, when the context graph is computed. These relative frequencies are computed in the context of a clique, by taking into account all its nodes.

There are multiple ways of choosing a proper valence for a word from all the possibilities offered by SWN, but the *dominant* function can be always computed. Moreover, we suppose that the clique is non conflictual if:

1. the *dominant* is unique, i.e. there is no other side having the same dominant value
2. the value of the *dominant* is more than 0.1, which means that there is at least a significant difference between the positive, negative or neutral side

The conflict can be solved in multiple ways and we propose three methods for this purpose:

- a method based on maximum valence selection (**SWN Max**)
- an heuristic based on a greedy dominant decision (**Contextonym Average**)
- a dispersion minimization method (**Contextonym Optimized**)

SWN Max resolves the conflicts by choosing the maximum valence for each WordNet synset. This method is similar to the proposition of C. Potts [156], with the difference that equal valences (opposite or not) are not treated as neutral. In fact, for equal opposite valences (very few cases) we take both sides as reference.

Contextonym Average This method is a greedy dominant selection and consists in two different strategies: first is the choice of a dominant side for this clique (positive, negative or neutral) and second, the selection of a valence for a selected node.

For the dominant decision, we assume that every node has a maximal valence potential. In this scenario, for the dominant computation, we always choose the maximum

absolute valence from the list associated to the node. If two opposite options exist, we choose the valence with more occurrences.

After the dominant is computed and a side has been fixed for each clique, for each node, only the valence corresponding to the given side is chosen. If more than one such valence exist, we compute the average of the existing valences and report it as the selected value.

Contextonym Optimized For our second approach, we decided to explore every possible choice for a valence. Since the hypothesis of a strong dominant for every clique has been made, this idea suggests the choice of similar valences for every node. This similarity is measured in terms of dispersion across the clique and it is given by the equation 3.13:

$$disp = \sum_{w_1, w_2 \in \mathcal{Q}} freq(w_1) \times |valence(w_1) - valence(w_2)| \tag{3.13}$$

This method minimizes the dispersion for any valid solution. A solution is considered valid if a strong dominant can be computed with the valences chosen by this algorithm. By using this method, no dominant could be computed for 12 cliques, for which the valences have been manually decided.

By introducing the context (**Contextonym Average** and **Contextonym Optimized**), we manage to solve all the SWN ambiguities and inconsistencies, which is the first of our major contributions. We do not manage to cover all the words from the discrete dictionaries, because of their rarity in our corpus.

3.4.4 Validation

For the validation we decided to make a similar setup as C. Potts [156] did for his disagreement experiment. We compare the valences from a set of well known affective dictionaries to SWN or our proposed models. Due to the fact that all the dictionaries used in the comparison contain only labels for the valence (positive or negative), we also transform the SWN valence into discrete labels. The first dictionary we choose is the Opinion Lexicon [112], followed by the Harvard General Inquirer Lexicon [186] and MPQA [207].

In order to compare all these linguistic resources, we propose a simple *Overlap* computed between our models and the lexicons. The first is an **SWN Overlap** computed on the SWN dictionary. If a word has already a conflict in SWN, than it is reported as a disagreement without regarding the valence in the other dictionary.

The **SWN Max** overlap computes a disagreement rate between the maximum valence for each word and the discrete valence found in the lexicon. This is based on our conflict resolution strategy, similar to the proposition of C. Potts [156].

Contextonym Average and **Contextonym Optimized** are contextual models, so an agreement is reported if there exist a context where the valence from the lexicon is found. Otherwise a disagreement is reported. These correspond to our conflict solving

methods described in the previous section.

The ***not found*** rate is represented by the number of words found in the lexicon which cannot be found in SWN or our contextonym model.

Opinion Lexicon (OL) [112] contains discrete manual annotations for positive and negative words. Moreover, it contains several misspellings for frequently used words. Figure 3.7 presents all the disagreement and *not found* rates, for the four strategies.

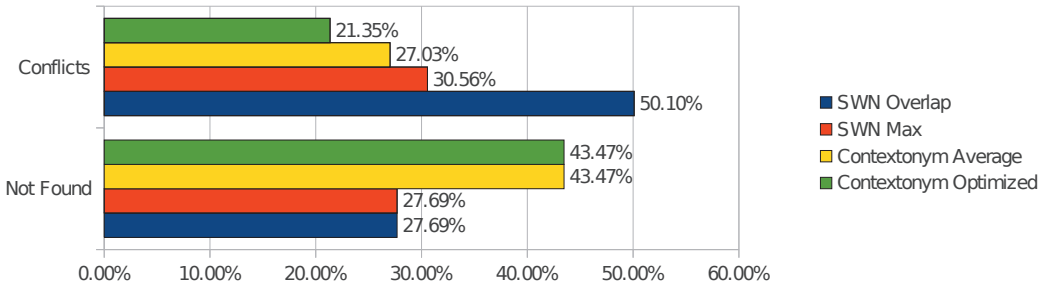


Figure 3.7: The Opinion Lexicon (OL) [112] disagreement and *not found* rates in comparison to different strategies for SWN corrections

MPQA Subjectivity Lexicon [207] contains annotations based on the polarity (positive or negative). Figure 3.8 presents a comparative diagram of all these results. The overlaps are computed in the same way as for the Opinion Lexicons.

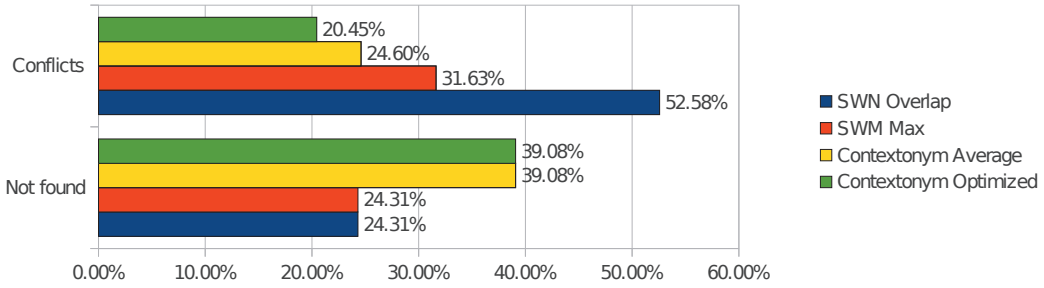


Figure 3.8: The MPQA Lexicon [207] disagreement and *not found* rates in comparison to different strategies for SWN corrections

Harvard General Inquirer (HGI) [186] is a lexical resource which is concentrated in attaching semantic information to the words. It contains positive and negative labels for most of its words. The overlaps are computed in the same way as for the previous lexicons. Figure 3.9 presents a comparative diagram of all these results.

The original disagreement rates are quite high (50.10% for OL, 52.57% for MPQA and 78.16% for HGI) and they are mainly related to the original conflicts found in SWN dictionary. Computing the maximum of each valence set decreases the disagreement (30.56% for OL, 31.63% for MPQA and 41.45% for HGI). It is not clear how the context of the word is chosen by this method, since the maximal valence of a synset does not necessarily correspond to the most frequent meaning of it.

The Contextonym Average method uses an heuristic similar to the one proposed by

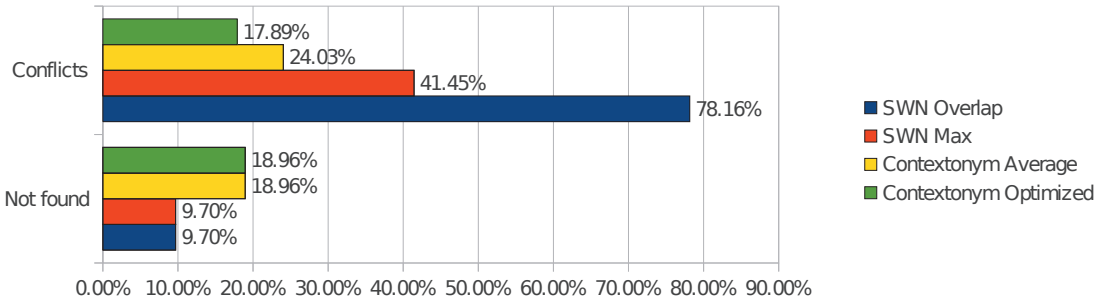


Figure 3.9: The Harvard General Inquirer (HGI) Lexicon [186] disagreement and *not found* rates in comparison to different strategies for SWN corrections

the maximum method: the dominant is fixed by the maximum. The main difference is that the choice of a unique valence for any word present in a clique is done in a contextualized way. This leads to the discovery of a new context for certain words, which further decreases the disagreements (27.03% for OL, 24.60% for MPQA and 24.03% for HGI).

Our last method manages to provide a unified algorithm for the choice of the dominant and the valence associated to each word. Moreover, the usage of the Contextonym Optimized method makes the valence selection consistent inside the clique, because they were chosen to minimize the dispersion. This leads to the lowest disagreement computed among the methods proposed: 21.35% for OL, 20.45% for MPQA and 17.89% for HGI, which is less than half of the initial amount.

The *not found* rate for the **SWN Overlap** and **SWN Max** cases is the same, since the metrics are based on the same dictionary: the raw form of SWN. For **Contextonym Average** and **Contextonym Optimised**, the *not found* rate is higher than in the case of previous two methods, but this is not surprising since very low occurring words were filtered out from our corpus. In our preprocessing phase, we applied a node filtering step which preserved only words whose frequency is higher than 0.01% and word co-occurrences with a frequency higher than 0.01 %. Therefore, the *not found* rate could be improved by adding more words with low frequencies in our model. Another aspect of this problem is linked to the style of our corpus. The subtitles are linked to dialogue style, whereas OL, MPQA and HGI contain words that are not only used in dialogue, but also in writing. Moreover, OL contains several common word misspellings, which are preserved in our comparison. Since we selected only the high quality subtitles, we expect a very low frequency of misspellings in our corpus.

In the proposed approach, the context is modelled in a simple but very effective way. Moreover, with our Contextonyms Optimized, we managed to find an automatic method for the context and valence selection. For the HGI case, we managed to decrease the number of conflictual words from 8 out of 10 down to 2.

3.5 Discussion

In our approach, the context is a key part of the solution for the disambiguation problem. We model the context by using graph-based structures and we extract the strong-

context (word co-occurrences with high frequency) by modelling it into contextonyms. By using these approaches, the disambiguation process is easy and more natural than in any previous work. Moreover, we model context as a contextonym (cliques within a WordNet) graph, which add a new semantic level to WordNet.

A new method for clique exploration called DDMCE is proposed, dedicated to large graph structures. Moreover, the DDMCE algorithm takes into account dynamic graph structures, very useful for practical problems. The platform offers a parallel approach, which can be easily distributed, based on an improvement of the classic exploration algorithm described by Bron & Kerbosch's algorithm ([24]) with Tomita et al.'s pivot selection strategy ([196]). Using a tree representation of our solution, the algorithm offers a high re-usability of the solutions collected, which gives, in the context of highly distributed architectures, more robustness to the algorithm. Moreover, DDMCE also introduces a set of heuristics to solve the data reduction problem, an open issue described in the Google Pregel article [118].

On the application level, even without the synset conflicts solved, SentiWordNet has been used successfully in Opinion Mining, Sentiment Extraction and Affective Feedback. Therefore, providing a model that solves some of these conflicts is a great achievement. The contextonym model is also dependent on the style of the corpus used to build it. This issue could be overcome by creating multiple contextonym graphs, one for each writing style. Moreover, due to the current style of our model, it can be used to detect user's affective feedback in interactive systems. To achieve this, a corpus based on a highly interactive scenario needs to be collected. For this purpose, we propose the methodology described in the next chapter.

Part II

Affective Interactive Systems

CHAPTER 4

Affective interaction

"Truly wonderful the mind of a child is."

– YODA, Star Wars Episode II

Contents

4.1 Context 86
4.1.1 Wizard of Oz experiments and avatars 86
4.1.2 Corpus collection 87
4.1.3 ACAMODIA Project 87
4.2 Related Work 87
4.3 Scenario 89
4.4 OAK 90
4.4.1 Scenario formalisation 91
4.4.2 Architecture 92
4.5 Project results 93
4.6 Discussion 95

4.1 Context

Previously we approached the problem of Automatic Emotion Detection, by proposing several techniques. Our final goal is to integrate all these algorithms into one unified platform that deals with rich interaction data. This can be approached with various methodologies and we propose a method to collect a corpus of interactive data, while building an innovative story telling environment for children.

Building a virtual natural environment, in which the participants can interact without any difficulty, is very challenging. Moreover, introducing a virtual conversational agent into this kind of environment increase the expectations of the human participants, up to the point where they can be disappointed by the agent's capabilities [127]. Building such an environment for children is even more difficult. Providing them a familiar environment, with natural reactions from a conversational agent, becomes critical.

Our purpose is to create a new environment, centred around the story telling activity, which allows all the participants to act naturally, even if the new dialogue partners are not their usual ones. This setup has two types of participants: a listener (the child) and a storyteller. The narrator (storyteller), can be either a psychologist present in a video conference mode or an avatar (an animated virtual character, driven by a psychologist). During the activity, the child is interacting with one of the partners for the first half of the story and it continues with the other. Our goal is to compare the two situations and to measure the difference between the two interaction environments. Moreover, the affective feedback is also important for our experiments. Observing the children while interacting with an affective virtual character represents one of our secondary goals. This is done by setting up a Wizard of Oz scenario, in the context of a storytelling activity.

4.1.1 Wizard of Oz experiments and avatars

Wizard of Oz (WOz) is a method used in psychological studies, human interaction or linguistics. During the design process, obtaining some early feedback on the model is crucial. Therefore, WOz experiments offer the opportunity to overcome the initial issues of the design, by introducing a new actor into the experiment ("the wizard" or "the pilot") which has the role of managing the system, while giving the sentiment of an artificial intelligence. In dialogue systems, these initial issues are related to the poor speech recognition for open dialogue or dialogue management. The WOz paradigm enables us to re-create a natural dialogue environment and to interact with the subjects.

In the study of human-computer interaction, the method of iterative design and bootstrapping dialogue models is very popular. For instance, Rieser et al. [162] used a Reinforcement Learning technique to train a dialogue model from few examples collected using a WOz method. The modalities used to collect the data differ from one experiment to another, but all have the same basic idea of a pilot driving the activity instead of an automatic system.

4.1.2 Corpus collection

In order to design interaction models, some research groups [6] proceed first into a collection phase, done through a Wizard of Oz perspective. Usually, this phase consists into getting Human-Human or Human-Computer interaction data, followed by an annotation and pattern extraction phase, which leads to an interaction model[1].

Setting up an experiment for this kind of corpus collection is usually very time consuming, particularly during the annotation phase after the experiment is done. In some situations, important notes observed during the experiment could be lost, if they are not annotated at the right time. The approach we propose requires a basic dialogue model, with some observable interaction states, to be built before the experiment starts. Moreover, it provides automatic annotation for the collected data.

4.1.3 ACAMODIA Project

The ACAMODIA project is a French PEPS project supported by the Institute for Humanities and Social Sciences (INSH) and Institute for Information Sciences and Technologies (INS2I) at French National Center for Scientific Research (CNRS). The objective of this project is to build a familiar environment, centred around the storytelling activity, that allows to collect rich data from all the participants. The most important actor is the child, since the project studies his reactions with a new conversation partner (a virtual character or adult in video conference). Nevertheless, the performance of the psychologist (in video conference mode or while driving the virtual character in WOz mode) can be studied as well. The result of this project is linked both to Computer Science by developing an interaction model based on the data collected, and to Psychology, by refining the current theories dealing with child-machine interaction.

From the technical perspective, the project needs to cover several *requirements*:

1. Scenario development: which includes an interesting story to be developed in the story telling environment and the protocol formalisation.

2. Multi-modal data collection: a new platform has to be developed in order to sustain the data collection infrastructure and to enable rich interaction between a child and the narrator.

3. Data Analysis: which is done from both psychological and computer science perspectives.

4.2 Related Work

Most of the current experiments done in the WOz perspective are not reusable [168, 48, 140], because of their strict link to the experimental set-up. Moreover, the interaction modality or the protocol is not the same in all cases. DiaWOZ-II [16] proposes

[1]In this chapter, the terms interaction and dialogue are interchangeable, since we refer to interaction models that are only linked with dialogue (verbal or non-verbal)

a simple text based interface used in tutoring studies for engineering and mathematics, whereas Whittaker et al. [205] use a web-based interface to simulate dialogues in a restaurant scenario. Based on the same idea of simple text interfaces with complex dialogue management, Munteanu et al. [129] propose a state based dialogue management prototype, with the possibility to introduce real-time new states into the model. Some early multi-modal interfaces, SUEDE [102] and Artur [13], develop the interaction with new layers: simulated speech recognition and synthesis (Suede) or image describing the learning process (Artur).

From the embodiment perspective, Cassell [30] introduced the idea of face to face interaction with an animated avatar. Even if the level of details used to represent the virtual character are very high, the low conversational capabilities and the character's non-natural reactions, induce inefficient interaction between the human and the system. This phenomenon is called "the Uncanny Valley" [127]. Moreover, this type of behaviour affects the empathy of the users towards the agents [15]. To overcome all the issues, the agent has to respond to the user's frustration [101] and become more empathic [135, 157], emotional [154] and react at the right timing with a gesture or posture adapted to the situation [159].

The influence of the animated virtual character (conversational or non-conversational) on the human perception is formalised as the "persona effect". Pedagogical [128, 14] and game [158] studies show the existence of a link between the presence of a virtual character and the user's performance, whereas Miksatko et al. [122] conclude that no such impact exists. Grynszpan et al. [72] conducted a multi-modal study, through a Wizard of Oz perspective, that revealed high influence over the patient's performance for user with high functioning autism. The SEMAINE project also started with a WOz experiment [175], which lead to a simple interaction model integrated into the final release.

In the context of child interaction, the virtual character's influence has not been studied much. An experiment done by Oviatt [141] reveals that children from age 6 to 10 have less disfluencies in speech when talking to a "Jelly-fish" animated character than in direct communication with an adult. Moreover, children are highly attracted by their new conversation partner and they accept the engagement. This project concluded that, due to the high rate of disfluencies, misspellings and pauses, it is almost impossible to transcribe accurately the child speech with an automatic speech recognizer.

Ryokai et al. [166] conducted a study on the potential usage of an Embodied Conversational Agent (ECA), named Sam, on a children tutoring scenario. The task was to speed up the literacy learning process (reading and writing) through narration. In this work, Sam tells stories in a collaborative environment. The virtual character looks like a friend from pre-school, but tells stories in a way that models narrative skills very important for literacy. Results of this study demonstrated that children had a good social engagement with the ECA which allowed them to learn rapidly more linguistic features (i.e. new words or difficult linguistic constructions).

Similar to the ECA experiments, other are conducted using robots. The same level

of engagement is observed for children with autism [108, 163], in tutoring scenarios [98, 75] or to develop early cognition processes [200]. The potential of the two fields, with applications in education and narration, is similar. Testing a conversational or narrative approach with a robot is slightly more complex and expensive, therefore ECA are usually preferred due to their ease of use.

Storytelling environment, having an ECA as actor, has been built so far by only few research projects [64]. Our work uses the virtual character to build social engagement with a child. Compared to the previous work, our study proposes a formalized scenario and the "wizard" needs only to supervise its execution. Therefore, we designed a platform called the Online Annotation Toolkit (OAK) that suits all the requirements listed in the ACAMODIA Project section. Moreover, after the experiment is over, the data collected is already annotated with several observations.

4.3 Scenario

The challenge for the first phase is to find the proper setup for the experiment and a story to suit our needs. Several options can be considered: 1) an open dialogue setup; 2) a non-linear scenario, with a story adapted to each participant; 3) a fixed scenario with timings and gestures synchronised to the child's reactions. The first option is a very challenging setup due to the current transcription errors and dialogue management [141]. The second idea is easier to implement and was the first scenario prototype we created. The drawback of this method is that it requires multiple pilots to perfectly synchronise the story with the emotional feedback, gestures, speech management according to the child's actions.

The final choice is a fixed scenario, that allows free-context input, adapted to unpredicted situations. Moreover, to make the story more interactive, several communicative "errors" are included, in order to help the child to react. By using this simplified setup, a pilot is able to concentrate more on the child's reactions, rather than making an effort to "conduct" the scenario.

The story chosen for this experiment is "The lost ball" (fr: "Le ballon perché"), about a school boy who decided to play with his ball before entering in the class for courses. During this play, the ball is kicked on a roof. To make the things even worst, the boy and his friends decide to recover the ball by throwing a boot, a school bag and a scarf. As they are urged to enter into class, the ball is not recovered. At last, a huge storm arrives and blows all the things off the roof, enabling their recovery.

The first phase of the setup starts with the presentation of the experiment. All the questions related to the scenario and equipment are asked, in order to build a confidence relation with all the actors.

In figure 4.1 we present the selected scenario. One half of the story is told in avatar mode and the other half by the psychologist in the video conference mode. The story is presented as 15 image slides, which brings a new level of details to the narration. The first 7 slides are narrated by one storyteller (virtual character or psychologist) and for the 8 remaining slides the narrator is swapped. The scenario includes also three types

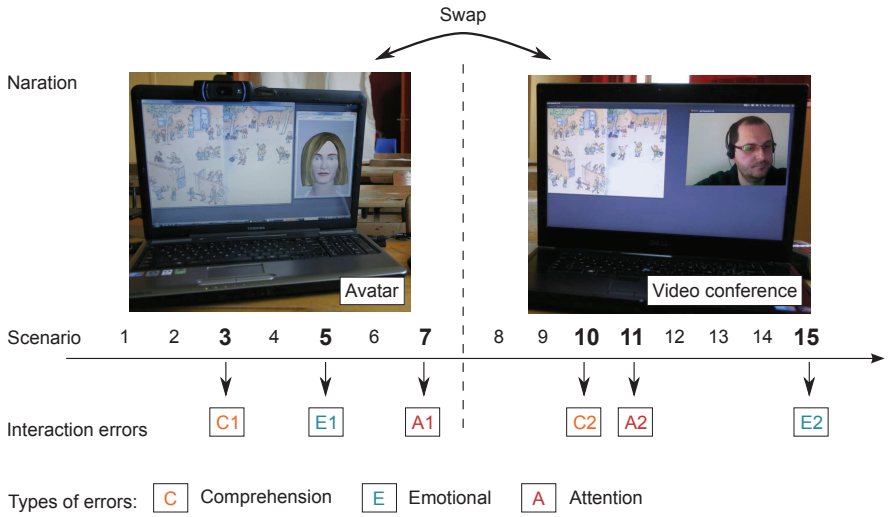


Figure 4.1: The ACAMODIA scenario, formalised into 15 slides and some of parts of the story have some communication errors included

of errors, in order to asses the attention (A1 and A2), emotional reflex (E1 and E2) or comprehension (C1 and C2). For example, during the C1 error, the narrator makes a semantic mistake by saying that "the boy throws his *carrot* on the roof", instead of saying *boot*. This type of error is used to test their attention to the details of the story. The emotional errors E1 and E2 represent a contradictory state of the scenario, where the agent simulates a negative emotion while speaking about a joyful event. This is meant to test the cognitive attention of the child, while observing the feedback over it.

At the end of the experiment, for each child, questions about these types of errors are included in a final survey. Moreover, they are asked to summarise the story and to detail all the "problems" found during the experiment. Half of the children started the narration in video conference mode and the other half with the virtual character, which enables the cross comparison of interaction during the two modalities.

More details about the protocol could be found in Bersoult's Master Report [17].

The model formalisation would follow the track proposed by Ales et al. [6], in order to produce dialogue patterns, which are automatically extracted from our data. The initial work has been done in Pauchet et al. [1] and consists in formalising a method to automatically extract pattern from manually annotated templates.

This protocol is integrated into the OAK model presented thereafter.

4.4 OAK

OAK unifies different platforms and concepts in a single tool, that is generic and simple enough to be used in real-time data collection, and requires simple manipulation skills. The way the avatar is driven has been simplified to the point where all the actions are very intuitive.

Another key point of the platform is the online annotation, given by the exact timestamp of the execution of a certain scenario item. This gives an idea about the order of execution of actions, the durations, and even formalises a trace of the interaction. It

has been used in real experiments with a good satisfaction level obtained by both the pilot and the children. Moreover, the collection of useful annotated data is simplified at the end of the experiment.

To demonstrate the genericness of the OAK toolkit, we describe the scenario formalisation and architecture on another interaction scenario, different from the one described previously.

4.4.1 Scenario formalisation

We define the formal concept of **scenario** in OAK as a finite state automaton compound of a set of **states** and a set of observations. The **states** are actions that are executed by the engine or translated directly into BML[2] [106] code. The **observations** correspond to elements of perception in the real-world, formalised as notes in the OAK scenario and from which the execution schedule is built.

Whereas the usage of the states is self-explanatory, the observations are used to maintain a certain logic in the scenario. They are not mandatory, but their usage is recommended to preserve a uniform level of execution for the scenario. Moreover, the usage of a state is logged with the timestamps of the appearance in the scenario. No formal or technical restriction for linking the states and observations are done, but it is highly recommended to keep the model clean and simple.

Figure 4.2 presents an example of scenario, with several states (s1-s3) and observations (o1-o5). As for experimental purpose, the transitions between these states are recorded, since there should be only one logical transition from a state to another, through the same observation.

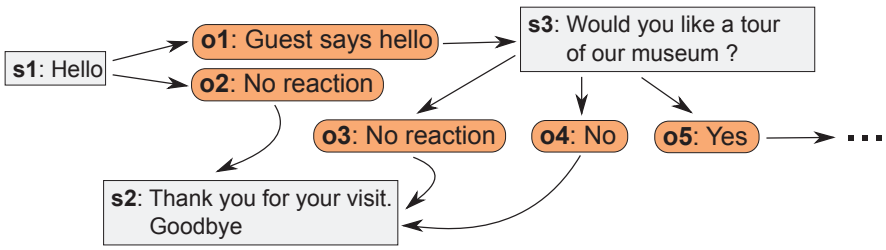


Figure 4.2: A simple example of a WOz scenario which can be used with OAK. The boxes represent states, whereas the rounded boxes are observations

When a state has two transitions leading to two independent states, for the same observation, the transition model is ambiguous. To keep the transition model clean enough to be implemented in a dialogue system, it is required that the transitions are not ambiguous.

Listing 4.1 presents a BML code of the action executed when the action "hello" (**s1** in Figure 4.2) is triggered. This code is specific to Greta Virtual Character [154] because of the execution backend, but it can be easily modified to suit any other virtual character or agent that supports BML. The first important part of the code is the *speech*

[2]The Behaviour Markup Language (BML) is an XML specific language describing verbal and non-verbal behaviour specific for humanoid virtual agents [106].

level where the actual speech is executed. The *face* level triggers the face animation of the avatar. In this example this is a face expression representing an intense emotion (happy-for, intensity=1.0).

```
<oz-bml>
  <bml>
    <speech id ="speech-1" start="0.0"
        language="english" voice="realspeech" text="Hello">
      <tm id="speech-1:tm1"/> Hello
    </speech>
    <face id="emo-1" start="0.0" end="3.87">
      <description level="1" type="gretabml">
        <reference>faceexp=happy-for</reference>
        <intensity>1.0</intensity>
      </description>
    </face>
  </bml>
</oz-bml>
```

Listing 4.1: The "Hello" BML action with code specific to Greta Virtual Character[154]

4.4.2 Architecture

Our system expands the current open source architecture of the SEMAINE Project [175], using the simulation part of this project, and embedding new components to gain full control over the architecture. Moreover, in order to test the child's adaptation to the narrator, a mixed setup (ECA and video conference) is done. The OAK system has three major components:

1. the Semaine Platform, which contains a component based communication system
2. the Greta Virtual Character [154], which is part of the SEMAINE project, and has been preserved in OAK. Potentially, it can be replaced by any other virtual avatar, agent or robot (such as NAO [64]) that interprets BML.
3. OAK, which consists in a pilot graphical interface (figure 4.3), and two views at the user level (figure 4.4)

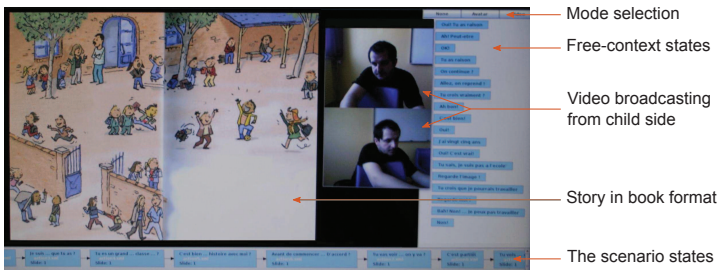


Figure 4.3: The pilot view of OAK

The first interface (Figure 4.3) is used by the pilot. It has a scenario area, which represents the collection of all the possible states. A state can be executed at any point, as many times as necessary. On the right, the free-context library is presented. It

consists in a set of states that are not directly linked to the context of the story, such as: “OK”, “You are right”, “Shall we continue ?”. On top of that library, a menu that allows the selection of the experiment mode is present: *none* (also known as start), *video* and *avatar*. The start mode corresponds to the beginning of the experiment, through the setup description phase. The other two modes correspond to the scenario split.

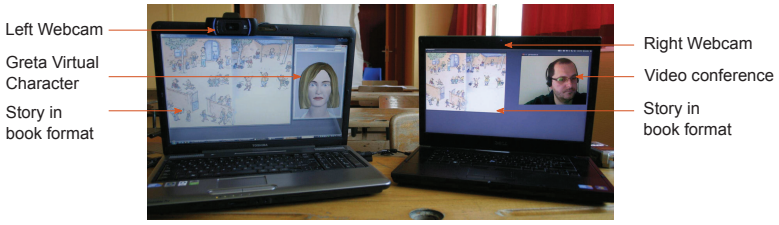


Figure 4.4: The two child views designed for OAK

The child interface (Figure 4.4) consists in two different views, one for the WOz part of the experiment and a second for video chat. On the left, the narrator role is played by the Greta Virtual Character, whereas on the right a video stream is used. This setup has two web cams, which allowed us to film the child from different angles. The video stream recovered is sent to the pilot view, as well. The video conference setup is done using multiple communication channels, built with the GStreammer [191] toolkit for Linux. All the videos recovered are saved on multiple copies, to ensure backup.

An important element present in all the three views is the story in book format. The images are digitized and synchronised among all the three views. Moreover, the pilot can use the mouse to point at important aspects of the story.

All the components of OAK are fully customizable, with independent XML based configuration files for each of them. The actions are translated into BML [106] code by an action interpreter and forwarded to the required agent.

Using all the data recovered through the experiment, we are able to conduct a brief statistical analysis based on several interaction features.

4.5 Project results

During the data collection phase, we managed to conduct the experiment on a valid population of 49 children, with ages varying from 6.4 to 9.3 years old, coming from 2 schools in the Rouen metropolitan area, France. The group selected for this analysis consists of 20 children (7 girls and 13 boys), which were chosen due to their age homogeneity and development, providing statistically relevant results, as well.

This analysis is extracted from the Master Report of Bersoult [17], which provides more details about the methodology and the psychological aspects of the issues. Our interest in this analysis is to provide a conclusion on the agent design aspect, which leads to the next generation of conversational agents dealing with children oriented models.

Figure 4.5 provides some quantitative measurements over the number of pauses (with more than 2 seconds of delay), sentences, words and interactive sentences. All these measures are summed for the entire population. The pauses, sentences and words

are the ones actually spoken by the child, while the interactive sentences are triggered by the psychologist, in video mode or while piloting the avatar, in order to make the conversation more natural and fluid. They consist in sentences that are taken out of the story context, such as: “Yes. You are right”, “OK”, “Do you think so ?” or “Do we continue ?”. The data is represented for both types of narrators: avatar and psychologist in video conference mode.

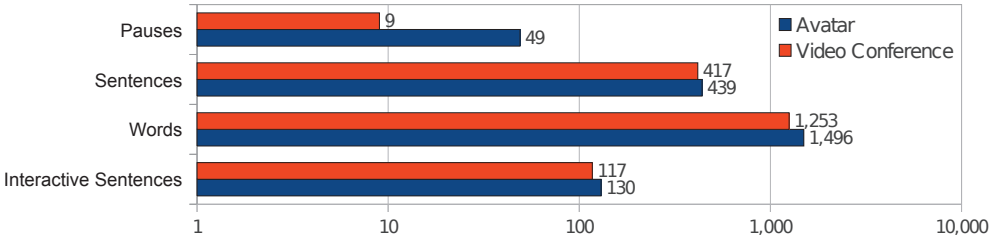


Figure 4.5: A quantitative analysis of the results, for the selected group of children

Except for the number of pauses, there is not statistical relevant difference between the two modalities (as the report of Bersoult [17] shows), which means that the children do not perceive an actual difference between the two modalities. The only significant difference is the number of pauses taken to respond to the questions. This cannot be linked with an attention deficit, because all the children participating on the experiment successfully responded to the final survey, which consists in describing some specific aspects of the story. We believe this could be linked with the style of the narrator, as the avatar tends to be more monotonous than the psychologist. Nevertheless, this offers a good feedback to the new system design.

The disfluencies[3] ratio is computed in a 100 words window. Figure 4.6 presents these results. Comparing the results we obtained to the one presented in the Oviatt study [141], the ratio between the two modalities is lower: 1.29, in comparison to the 2.5 (up to 3) presented by Oviatt. Several hypothesis could be made. First, when the avatar takes a human-like form, the disfluency ratio increase. Second, because of the video conference mode, the disfluency ratio is lower than it would be in the presence of a “real adult”.

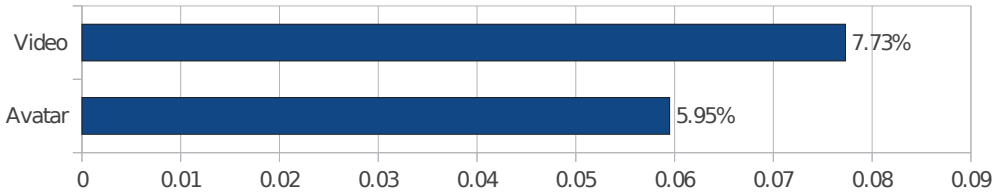


Figure 4.6: The disfluencies results, for the selected group of children

Based on the short survey conducted at the end of the experiment, we found several interesting comments regarding our experiments. First, the children were able to detect several differences in the appearance of the two narrators. Some of them observed the absence of a microphone and headset on Greta. Others compared the character with

[3]The disfluencies are markers of irregular speech, pauses and non-lexical vocabulary that would not be employed in a fluent dialogue

a toy or a "lady made of modelling paste" (plasticine). Moreover, 55% of the children detected the slowness in interaction with the character, but none of them were worried about this. In fact, all of the children adapted very well to the rhythm and appreciated that this allowed them to speak.

During the communicative error states, we observed an interesting difference in the interaction modality. The children use face gestures or postures more often to indicate that something went wrong when talking to the adult in video conference mode. While discussing with the virtual character, this tendency migrated to verbalisation, rather than using gestures. Moreover, during the interaction with the avatar, they used shorter and more concise sentences.

Based on the selected statistical results, we can conclude that children are able to adapt to the system, and that they enjoy the interaction with it, even if it is not as natural as with a real narrator. Moreover, due to our avatar interactivity, only very few children compared it with a cartoon like character.

4.6 Discussion

During the experiment, the children interacted with the virtual character similarly to how they interacted with a human in video conference mode. This can be explained by their high ability to adapt to this kind of systems and that they have lower expectations than the adults. This result sustains our initial hypothesis, that the children are engaged into the interaction with a virtual character. Therefore, in future, this offers the possibility to test new interactive models with children, using multiple modalities and emotions.

Building OAK allowed the psychologists to model the protocol and scenario very easy. Moreover, the selected results show that the children are able to adapt to the new environment well, without making any effort. The experiments show several difference in the interaction modality and a low level of disfluencies when the children are interacting with the virtual character.

In future, this system could be used to test the disfluency hypothesis in a more complex environment, involving adults, filmed or "in person", virtual characters or robots. Currently the platform is generic enough to allow the addition of other actors.

Furthermore, the interaction model recovered can be used to build an Intelligent Interactive Agent. The scenario and the protocol remain the same, but the agent needs to perceive the interaction clues (i.e. sentences, pauses, face gestures) and properly trigger the scenario state. A good Embodied Conversational Agent should be able to follow the scenario, already modelled, and interpret the user's feedback (affective, gesture or semantic) and respond accordingly. All these feedback channels are required for an interactive system, therefore a platform that allows an easy integration of such components becomes our next goal.

CHAPTER 5

AgentSlang: A new platform for rich interactive systems

"It is far better to adapt the technology to the user than to force the user to adapt to the technology."

– Larry Marine, Founder of Intuitive Design & Research

Contents

5.1 Context 98
5.2 Related Work 99
5.3 A new platform for Distributed Interactive Systems: MyBlock 104
5.3.1 The principles of a Distributed Interactive System 104
5.3.2 Description of the platform 105
5.3.3 Distributed aspects of MyBlock 108
5.3.4 Data oriented design 110
5.3.5 Components, Services and Schedulers 112
5.3.6 Example 114
5.3.7 Performance 116
5.4 Syn!bad 118
5.4.1 Context 119
5.4.2 Example 119
5.4.3 Implementation 120
5.5 AgentSlang 123
5.5.1 System components 123
5.5.2 Input and Output components 124
5.5.3 Natural Language Processing and Dialogue Models/Components 127
5.5.4 Affective Feedback 129
5.6 Discussion 131

5.1 Context

In the recent years the information becomes more available and it is constantly growing in size. Even in the interaction context, the human reactions in front of a computer are richer, they embrace multiple forms and factors (i.e. visual feedback, emotional, vocalisations, speech, dialogue), while dealing with multiple information sources. The primary conclusion given by the ACAMODIA Project[1] is that the Interactive Systems (IS) should be fast enough to react to all the input, because there is no significant difference between the reaction of a child in the presence of a computer or another human.

Dealing with rich interaction, given by multiple sources, in a very fast and reliable way has become our current challenge. The human-computer interaction problem is complex [6] and requires work on multiple levels, such as speech and feedback recognition, natural language processing, dialogue management or speech generation. Each of these problems has been treated independently, or at most a combination of the two, but to our knowledge, a system dealing with all of them has not been proposed yet.

The integration problem becomes more complex when a certain genericness is asked. In Ales et al. [6], we proposed a theoretical flow to solve this problem, but the idea of an implementation came later. In our initial approach, the flow requires speech input, which is translated into knowledge for the system and a dialogue manager should provide an answer to the query. This is synthesised into speech and simulates the communication.

In the current thesis, we add another level to our initial work represented by the affective feedback, as it has been described in the previous chapters. Figure 5.1 represents a description of a generic system architecture, as presented in Ales et al. [6], and having the Affective Feedback component in addition.

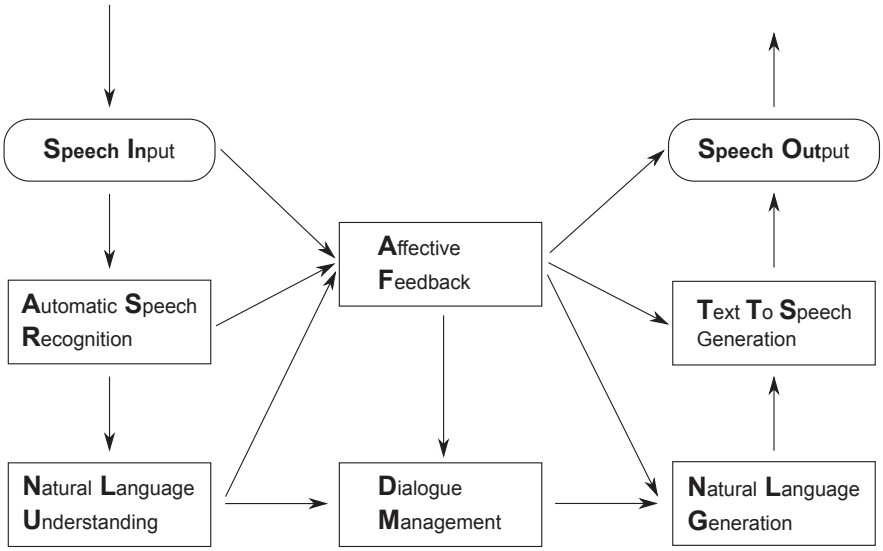


Figure 5.1: The general system view having a new layer that deals with affective feedback.

In general, the feedback is another communication channel that intersects sometimes the regular interaction feed. The feedback receives input from any component

that processes the raw input and offers data to all the components which refine the results. The system output needs to be adapted to the feedback at any level. The affective feedback, in particular, deals with multi-modal data: raw speech, transcribed data, gestures or even dialogue states. The output of such component is the degree of positivity or negativity present in the content in the case of a valence model, or a more complex system of labels in other cases, such as Ekman's six emotion model [55].

Due to the high numbers of connections between the components and the diversity of the inputs, a high speed architecture with dynamic links between the components is desired.

5.2 Related Work

An IS, as a complex system, contains multiple critical components. In Figure 5.1, we formalised all the components dealing with user inputs as Knowledge Extractors, the Dialogue Manager as an Interaction Manager, the components generating the output are Behaviour Generators and the ones executing this are the Players. For each one of these categories, the problem has been approached by multiple research projects. Our focus in this chapter is on platforms that propose architectures to model the integration of all these components into a Distributed Interactive System (DIS).

The components that interpret the behaviour, the Players, can be either a simple Speech Interface, an Embodied Conversational Agent (ECA) or a Robot, for instance. We focus the presentation of related projects on systems that do not have a strong link with the interpreters and are generic enough to be used in multiple environments and scenarios.

Mirage

Thorisson et al. [193] propose a constructionist methodology to build a component based architecture. This can be adapted to design ECA systems, by modelling reusable components that can be plugged in an unified architecture. The project aims at building an agent that is part of a collaborative environment, in a tutoring scenario. The agent acts as a tutor for presenting the laboratory material by using multiple modalities: speech, gesture. Moreover, the character is animated using head mounted 3d glasses and it is able to perceive gestures through a series of wireless sensors and speech.

The platforms aims at identifying existing software that could be reused and integrated into the project. The system is build against a blackboard messaging platform, Psyclone [38], which allows the integration of different components written in C++ or Java. The message format is a series of well documented, hard-coded, protocols, that are distributed across the network on multiple machines in order to obtain real-time performance. One of the challenges tackled by the project is the integration of multiple message sources providing similar data into an unified protocol.

This project described a good methodology for building component based architectures. The platform has no source code publicly released, except for the middleware

software (Psyclone [38]), which is available for research purpose.

GECA

The Generic Embodied Conversational Agent Framework (GECA) [81] aims at building ECAs that interact with human users in face-to-face conversations, while being able to recognize verbal and non-verbal input, generate speech, gesture or posture behaviour and perform basic conversational functions (i.e. utterance turn taking and feedback detection). All these functions are not domain specific, therefore Hung et al. [81] present this architecture as a generic platform that could serve as a foundation for future development.

The platform supports C++ and Java integration through the OpenAIR protocol, integrated in the latest revision of Psyclone [38]. The messages are sent using several distributed blackboards. The system proposed various formats for the data messages, such as standard existing XML formats. Moreover, the authors propose a generic markup language GECAML [81] which unifies all the features offered by various Automatic Speech Recogniser and Text to Speech API.

The application domain for this platform include: Dubrovnik (Croatia) city tour guide agent [83], application for experiencing cross-culture gesture difference is a virtual environment [82] or tutoring agents for question-answer scenarios [81].

VHMsg

The Virtual Human Messaging (VHMsg) [199] is a library that defines a message protocol around ActiveMQ, for building Distributed Interactive Systems. It was created at the University of Southern California's Institute for Creative Technologies. This project is part of the Virtual Human Toolkit[1].

The platform focuses on providing an easy solution for sending and receiving messages within the system. It provides support for various programming languages, such as : C/C++, C#, Java and TCL. Moreover, this library acts as a generic middleware solution and does not provide any support for binary data formats, XML or custom objects. The system management functions are limited to simple custom text messages.

Companions

Companions [31, 89] is an Embodied Conversation Agent (ECA), developed as an European Project by a consortium of universities and private companies. This project develops the idea of a companion [18] agent, which is engaged into a long term interaction process to build an empathic [150] relation with a human. The companion, as a role, is both a confident and a partner in daily interaction. This idea has been exploited by several groups in order to provide companions for medical purpose [189] or hospitalised children [167].

[1] `https://confluence.ict.usc.edu/display/VHTK`

Building empathy and trust between a human and a computer is a difficult task [150]. Therefore, the Companions ECA is build around the scenario *"How was your day ?"*, which acts both as a general interaction theme, with an open dialogue. The human is expected to provide any kind of informations regarding daily activities, while the agent acts as a listener and provides feedback and advices, when necessary.

From the technical perspective, the system uses several proprietary platforms, developed by the industrial partners: a middleware platform, Inamode, developed by Telefónica I+D[2]; an ASR and TTS engine, developed by Loquendo[3]; and a Virtual Character, developed by As An Angel[4]. Moreover, this system has two versions: one for English speakers [31] and a second for Slavic languages, with a study case for Czech language [89].

Unfortunately, due to the use of proprietary technology, the platform does not offer many research or technical details, such as validation, licence or performance aspects. Moreover, the ECA supports sessions of only 15 minutes long, due to some technical restrictions. Finally, the components of this platform and the algorithms produced by this project are not generic enough to be used in other environments.

Semaine

Semaine [175] is a Sensitive Artificial Listener (SAL), built around the idea of emotional interaction. It has been developed as European Project by a consortium of six laboratories and universities. The project is focused on the the idea of a Virtual Character that perceives human emotions through a multi-modal setup and replies to them accordingly. The response is not always a direct reaction to the affect perceived as a certain level of planning is done. Moreover, several virtual characters with different personalities are proposed, each having a different reactive model to the perceived emotion.

The affective models are proposed by a team of psychologists, which involves the proper tracing of human emotions [42] or the influence of emotions in every day interaction [41]. This collaboration lead to a multi-modal interaction corpus, collected through multiple sessions with dialogues between two humans playing different roles, and human-computer with the proposed system.

Originally, the project started as a collaboration between multiple partners through the HUMAINE [86] network, which lead to a proposition of a Emotion Markup Language standard (EmotionML [177]), developed to cover all the psychological and technical aspects of the problem. The Semaine consortium proposed another API standard, formatted as a markup language: SemaineML [175].

The system introduced the idea of component based, distributed interactive system, where each algorithm or component could act independently. The platform integrates various existing components into a distributed system that communicates over ActiveMQ [182] message queues. The affect detection part is a fusion of low level speech features extracted using OpenSMILE [60] and face gestures classified using iBug [183].

[2] `http://www.tid.es/en/`

[3] `http://www.loquendo.com/en/`

[4] On May 2013 the website of the company does not work any more: `http://www.asanangel.com/`

The behaviour of the agent is managed by two components developed by the team, which is send to the Text to Speech synthesiser: MaryTTS [144]. The speech is transmitted to a speech and gesture synthesis component, which converts the data into Greta BML code [154].

Due to the genericness of the platform, all the components are available individually. Moreover, except several components,the project is available for the research community as an Open Source project, under LGPL licence.

Others

The **MULTIPLATFORM** project [77] served as a component integration platform for various well known projects: Verbmobil [202] and Smartkom [203]. Herzog et al. [77] propose a semantic separation of the system functionalities into: recognition, analysis, action generation and synthesis. In order to facilitate the integration of various components, a new middleware layer has been added on top of the PVM platform [63], which introduced a publish-subscribe architecture. The project is not actively maintained any more and the source code of the platform is not available.

In the context of smart space applications, **CHILix** [46] proposes a platform where perceptual components used by audiovisual processing and multi-modal inputs are fused. The system uses a middleware approach to exchange hand crafted XML messages across multiple parts of the system. Moreover, the platform uses the freely available NIST[132] platform to collect sensors data across the space. The core idea of the project, the fusion of multiple sensor data seems appropriate for Ambient Intelligence applications, but the domain-specific messages and components are not generic enough to be reused in other scenarios. Moreover, the framework does not appear to be publicly available.

The **OpenInterface Platform** [110] aims at creating a reusable multi-modal environment for interactive systems. The components are described in Component Interface Description Language (CIDL), which has an XML-based syntax to describe each component interface. The behaviour of these components is described in Java, C++, C# or Matlab. All these layers are connected to a custom OI Kernel, used to route the local messages, whereas for remote connections, TCP and RCP interfaces are provided. The platform is available for research, but the latest release 0.4.0 (December 2009) is still in "alpha" state, without any other updates.

Summary

Table 5.1 presents a comparison of key features across main state of art systems. The core functions of our approach are the Dialogue Management and Affect Oriented design, based on specific components that support this claim. Currently, only Semaine and Companions projects use Affect Oriented features. The Dialogue Management is offered by Companions, Mirage and GECA. VHMsg is more a middleware library for building interactive agents, with no specific system implementations.

	Semaine	**Companions**	**Mirage**	**GECA**	**VHMsg**
Middleware Platform	ActiveMQ	Inamode	Psyclone	OpenAir	ActiveMQ
Integration Approach	pub/sub	plain socket	blackboard	blackboard	pub/sub
Operating Systems	LMW[1]	Unknown	LMW[1]	Unknown	LMW[1]
Data Interface	multiple[2]	XML	string	XML	string
System Management	Yes	(No)	No	No	No
System Events	No	(No)	No	No	No
Actively Maintained	Yes	Unknown	(No)	(Yes)	(Yes)
Platform Licence	LGPL	Proprietary	Unknown	Unknown	LGPL
Dialogue Management	No	Yes	(Yes)	Yes	No
Affect Oriented	Yes	Yes	No	No	No

[1]Linux, Mac, Windows

[2]String, XML and binary data converted to string

Table 5.1: A comparison of key features of existing State of Art Interactive Systems

Semaine is a Sensitive Artificial Listener, where the sensitive part refers to the affect recognition and simulation aspect. The listening key word refers to the ability to perceive certain emotions, but due to the lack of dialogue/interaction management, the agent is not able to reply with semantically adequate behaviour to the human companion.

Companions offers both dialogue management and affective interaction, but with very few details about this aspect. In fact, due to the proprietary licence of the system, no component or source code has been publicly released. This is making very difficult an evaluation of the quality of these features. Moreover, the system is centred around the *"How was your day ?"* scenario, which restricts even more the field of application.

Mirage is used for an augmented reality application, where the agent is participating to the environment. The platform allows the collection of multiple perception layers (i.e. gestures, speech), with the possibility to fusion. The dialogue capabilities of the system are very basic, by providing some reactive model linked with the perception.

GECA is a generic platform for ECA Applications. It does not offer support for affect oriented design, but is has basic dialogue models support. It uses a distributed approach in order to integrate different legacy algorithms in one unified platform.

Both Mirage and GECA use different versions of Psyclone (OpenAir is the previous version of Psyclone), which are not actively maintained. This makes very difficult the support of the projects, due to the use of unsupported middleware. On the system

management aspect, currently only Semaine provides tools for such a purpose, whereas the real-time system messages, that provide informations about the state and status of all the components, are not supported by any of the presented platforms.

5.3 A new platform for Distributed Interactive Systems: MyBlock

5.3.1 The principles of a Distributed Interactive System

Schröder [176], proposed a set principles that serve as a guideline for any DIS, and was the foundation of the SEMAINE API [175]:

1. **Distributed component oriented design**, which assumes that rather than building a monolithic system, which takes one instance of the data and combine multiple levels of processing in a single instance, a multi-component system is more useful because of the possibility to integrate existing components and convert data according to each component needs.

2. **Component reusability**, which makes the hypothesis that every existing algorithm or component, solving a particular problem needs to be easily integrated into the new architecture and all the components integrated need to be reusable as well.

3. **Scalability** is one of the most important features of a new platform since it would be desired to deal with large amounts of data and components

4. **Open Source Base Code** offers the possibility for all the community to integrate and benchmark their own algorithms on a common platform.

Our model introduces three new principles to this guideline:

5. **Data oriented design** refers to the system ability to process certain types of data, rather than concentrating on the data source. Our principle says that if a certain system, or more specific a component, knows how to process a specific type of data, it is mandatory to process it, and the source of that data is not important, as all the components in the processing pipe would make a "best effort" approach to transform the data.

6. **Fast, reliable components and total distribution.** Since the ACAMODIA experiments, we are expecting large quantities of data passing through our system, so speed and reliability becomes a critical requirement. Moreover, to eliminate any unneeded transfer between components, a broker-less architecture is desired.

7. **Dynamic component linking**, which refers to the ability of the system to build processing pipes at any point (runtime or deployment), without affecting the flow.

In order to incorporate all these principles in our work, and more specific the Component reusability requirement, we decided to "separate" the project into two distinct parts: MyBlock, which deals with platform issues, data and component distribution; and AgentSlang which is an instance of MyBlock: a repository of components build around MyBlock architecture.

Licence

Except the SEMAINE Project [175], all the other relevant project dealing with similar issues have licensing problems, by either being too restrictive or not considering a licensing system at all.

In order to assure the reusability principle, all the components of the MyBlock platform are licensed under The GNU Lesser General Public License v3(LGPL) and the French CEA CNRS INRIA Logiciel Libre (CECILL-C). This combination allows unlimited access to the project, with the obligation to publish under the same license any modification made on the project (LGPL) and any academic or commercial usage should be cited (CeCILL-C).

The licensing process for AgentSlang is more complicated, due to the possibility to integrate proprietary software or other licenses into the project. The final decision is that all the components published under the AgentSlang project should have the same license as MyBlock, until stated otherwise, for a particular component. The possibility to integrate multiple licenses is allowed due to the distributed architecture and the component independence, which can be considered in terms of free software, soft-binding.

5.3.2 Description of the platform

The need for a fast and reliable DIS lead to the design of the MyBlock platform. It incorporates all the strong features of SEMAINE, such as component distribution, data independence and component integration; but provides extra features such as speed, robustness, dynamic component linking, data-oriented design.

MyBlock is designed to be a middleware for distributed pipeline processing. It had to be small, flexible and fast enough to support the development of a rich platform, capable of exchanging information in real-time. The main platform has a three level separation, like in most of the modern architectures, which allows an easy management and understanding of the components. Moreover, each level is responsible of a different function. Figure 5.2 presents the three levels of MyBlock, separated by their function. Each level has a different set of keywords associated to facilitate the understanding of the concepts associated with that level.

Components

A component is an atomic structure for the MyBlock platform. The component processes a given set of data types and forward the output to the next component in the

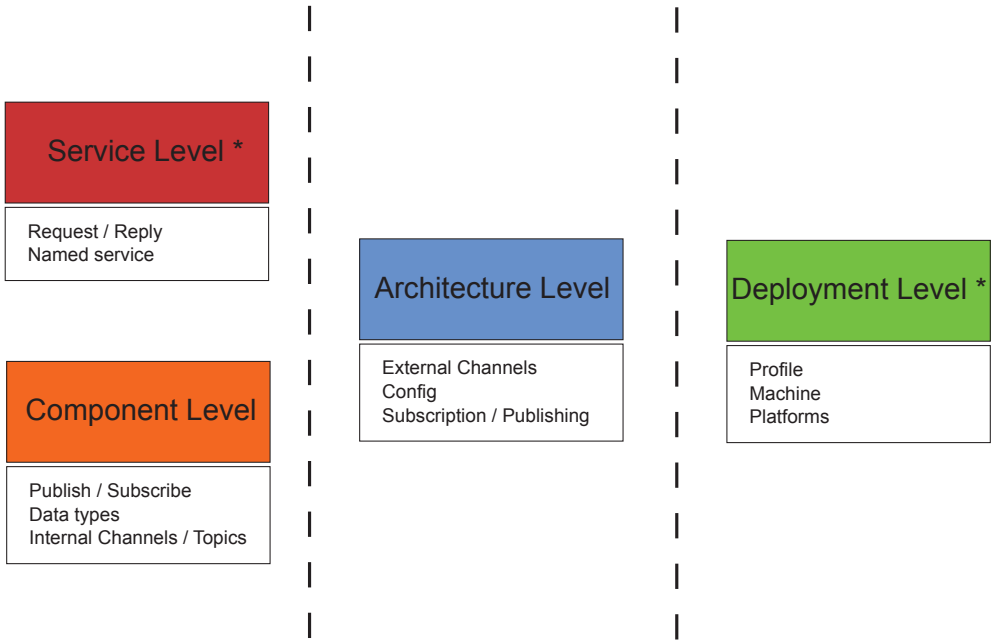


Figure 5.2: The MyBlock functional separation is done in three different levels, each having assigned a different set of responsibilities. Each level has a set of key words assigned, which describe the functions of that level. The levels marked with * are more linked to advanced concepts and would not make the object of a simple integration task.

chain. The internal flow of a component can have two different aspects: either it is a reactive output to the input, or an active component, in which case it can produce output based on the internal states, without having an input. A special case of components consists in elements which only consume (Sink) or produce (Source), without having a mixed function. In theory, these two types never exist, because every component has a mixed role. For a more practical approach, we consider sources to be the elements which only provide data to the MyBlock platform, even if the component is just a proxy for another source (a microphone), and sinks to be elements that just consume, even if their function requires redirecting the information to another sink (a set of speakers, for instance).

At this level, the data types are an important aspect. The data exchanged between components need to be compatible between linked elements. A component formally defines the preconditions and postconditions in terms of data types, which allows the linking with the other components to form complex processing pipes. The communication protocol between two components is a simple publish-subscribe architecture, which allows an easy exchange when data is available.

To allow the binding of different elements into a processing pipeline, the components define a set of internal channels, to which it publishes the data. These channels are identifiers for each function fulfilled by the component and could correspond to one or multiple published data types. In general, it is a good practice to publish only compatible data types on the same channel. In order to preserve the terminology with other distributed message processing systems, these channels are named *topics*.

Services

At the first level of the MyBlock architecture, the same as the component, we define another very important concept, with a different functionality: the service. Its main function is to respond to requests that can be triggered by any component or service. The communication protocol used for service is a synchronous request-reply, which allows a simplified logic for the knowledge query. When making a request, an element instantiates a client, which makes the request to the selected service. Each service has an unique identifier, given by its function, which allows a simple localisation.

Writing a service is considered an advanced topic mainly because most of the issues on IS can be resolved by a component pipeline.

Architecture

On the architecture level, the actual pipeline is constructed, by passing the configuration parameters to the components and services and building the dynamic links between all the elements. The important features of this level are highlighted by the ability to change dynamically the structure of the processing pipeline, without changing the logic of the components. If the data exchanged between several elements is compatible, the order of processing is not important. Moreover, in comparison with other platforms which use the processing pipeline paradigm, MyBlock does not need special components for data multiplication or joining. This is automatically ensured by the construction of the platform.

Deployment

The deployment level is essential for the production-ready environments. While building and testing the platform, all the deployment aspects of the system can be ignored, but at the end, migrating from a development stage to production should be as direct as possible. The first property of this level is that all the elements (services or components) can be grouped into profiles. This corresponds to the ability to group components with similar functionality under the same profile, which makes management and understanding easier.

One or multiple profiles can run on the same physical machine. The only real restriction is that a profile group cannot be split across multiple machines. All the profiles found on all the available machines are grouped into one large project setup, which gives a global view over the structure of the system.

The deployment level corresponds to the **Scalability** (item 3, page 104) and **Dynamic Component Linking** (item 7, page 104) design principles presented in the previous section. In general, the components do not need to be aware of deployment architecture and all the issues concerning these principles should be resolved by the platform.

5.3.3 Distributed aspects of MyBlock

Based on the platform design principles enumerated in previous section (page 104), having a platform that is fast, reliable and fully distributed becomes a critical point in our design. This requires a middleware platform to manage all the communication between the components. This platform needs to be able to deal with distributed environments, be actively maintained and model a simple communication protocol. Following these requirements, several candidates can be considered. Table 5.1 highlights several candidates: ActiveMQ, Inamode, Psyclone and OpenAir (Psyclone). Inamode is a closed source proprietary platform which cannot be tested, therefore the benchmark needs to be conducted between ActiveMQ, Psyclone and our proposition.

MyBlocks has a level of abstraction for the transport layer, that we implemented using ZeroMQ [79]. ZeroMQ (also spelled ØMQ) is not a typical middleware platform, as other projects use, but an intelligent transport layer which unifies different networking protocols (TCP, UDP) among with in-process and inter-process communication. Moreover, due to the reimplementation of the classical socket api, the ZeroMQ library is proved to be more efficient in massive distributed environments [79]. Due to the limited support of in-process and inter-process protocols and the recommendation of ZeroMQ authors to avoid the usage of UDP as much as possible, our choice was to use the TCP protocol.

ZeroMQ: basic principles

One of the advantages of using ZeroMQ as transport layer is that it already implements several communication patterns. These pattern are linked with the communication topology and the behaviour of the sockets. The three main topological configurations offered by this library are:

- **Request-reply**, allows the connection of multiple clients to several servers in a query-response setup. It is useful in a remote procedure call or query instantiation.
- **Publish-subscribe**, allows an easy data distribution from a set of publishers to a group of subscribers
- **Pipeline**, which connects multiple elements into a multiple steps setup. This can be used for parallel task distribution

Due to the flexibility offered by the publish-subscribe pattern and the protocol support, our choice was to use this rather than the pipeline protocol. For the services and clients architecture, we decided to use the request-reply pattern, since it is a classical remote procedure call.

Benchmark

Following the previous list of candidates for middleware platforms, we decided to perform a benchmark test between ActiveMQ [182], one of the most representative systems

on the Message Queue (MQ) implementation, and ZeroMQ. The other candidate for this test would be Psyclone [38], one of the very popular white-board style message systems, but Schröder [176] already conducted a benchmark and concluded that ActiveMQ has a huge advantage over Psyclone.

The machine used for this test is a I7 Intel machine, at 1.6 GHz per core and 3.9 Gb of RAM. The operating system used for the test is an Ubuntu 12.04 Linux, with Oracle Java 1.7.

Table 5.2 presents the running times, in milliseconds, for ActiveMQ and ZeroMQ and table 5.3 shows the message throughput. The setup for these times is to send a series of random messages, for a given size, from one component to another. We choose to send random sequences in order to prevent the caching speed of ActiveMQ, which applies a set of heuristics in case the same message is sent over the network. Moreover, in order to prevent local traffic peaks, we sent 100 messages and presented only the average time. For the message throughput, we represented the number of messages that pass between the two components in one second. Figure 5.3 represents this measure, side-by-side, on a logarithmic scale.

System	**Message Size**					
	10	100	1,000	10,000	100,000	1,000,000
ActiveMQ	0.25	0.21	0.17	0.40	3.41	28.40
ZeroMQ	0.02	0.12	0.06	0.31	1.79	15.12

Table 5.2: A running time comparison between ActiveMQ and ZeroMQ platforms. The time presented is expressed in milliseconds and the message size represents the length of the sequence sent over the platform

System	**Message Size**					
	10	100	1,000	10,000	100,000	1,000,000
ActiveMQ	4,047	4,792	5,714	2,479	292	35
ZeroMQ	50,000	8,333	16,666	3,225	558	66

Table 5.3: A message throughput comparison between ActiveMQ and ZeroMQ platforms. The throughput presented is expressed in number of message per second and the message size represents the length of the sequence sent over the platform

The conclusion of this experiment is that currently ZeroMQ is a much better solution for message exchange, mainly due to the fully distributed state, the possibility to send binary data and the absence of a broker, which slows down a distributed pipeline architecture.

Distributed processing pipes

The alternative to a distributed pipeline is a monolithic algorithm, with all the procedure calls embedded into a large system. Such a proposition has the speed advantage, since

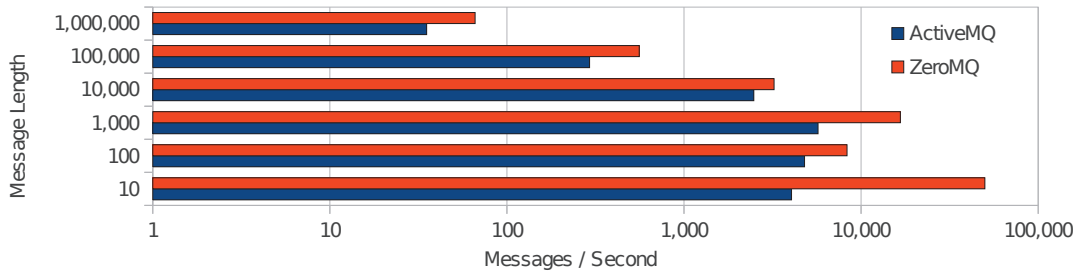


Figure 5.3: The performance comparison for ZeroMQ and ActiveMQ, for message throughput representation. The representation is done on a logarithmic scale

all the components pass just a data reference between them, rather than wrapping the data into different formats. The downside of this approach is the maintenance of this system, since it easily becomes almost impossible to comply every algorithm to the exact same data structure and transform any new functionality into an internal procedure call. Moreover, in the early stages of prototype creation, dealing with unstable procedures could make the whole testing process more difficult.

The main advantages of a distributed processing pipeline for IS are the ease of integration for new components, easy deployment and clear separation of concepts. In such of an environment, it is easier to establish a common protocol for data, which does not have to have the same strict format to be exchanged between components. In a distributed environment, each component is associated with a functionality and a couple of these elements could correspond to a sub-procedure call. All these concepts are fully separated and maintaining them is easy since the development is done for a component at the time.

In a production environment, for a monolithic algorithm, changing the setup or the order of procedure calls would mean, probably, a recompilation of the software. In distributed environments, this can be realised only by changing the architecture of the processing pipelines, which is much more feasible than recompiling.

5.3.4 Data oriented design

In a distributed pipeline system, such as MyBlock, the data representation becomes critical. Even if the formats do not have to be strictly identical between linked components, but at least the compatibility has to be ensured. There are two main directions in this area:

- Design data to have a small transfer size and memory footprint
- Generic data representation, written in standard formats (i.e. JSON or XML)

The ad-hoc feature representation is very popular in early system integration, but since no specification is used, the data becomes very difficult to be maintained. An alternative to this process is represented by Google Protocol Buffers [67], which formalises all the data messages into a strict syntax which is translated into messages and data types in various programming languages. This approach seems to be secure and flexible

enough for the usual data exchange between services, but it is very strict with data type inheritance, a concept supported by all the major Object Oriented programming languages.

The generic data formats have been formalised in the recent years, due to the increase in popularity of Semantic Web technologies. Due to increasing popularity of establishing strict interchangeable formats which a web service would "understand", several formats have been proposed. World Wide Web Consortium (W3C) is the authority that deals with current and future web standards, including the current web service data formats for the Semantic Web. Table 5.4 presents the current situation of various data type standards related to the conversational agent problem.

Standard	**Functionality**	**W3 Status**
VoiceML [87]	Automatic Speech Recognition	W3C Recommendation
SSML [26]	Text-To-Speech	W3C Recommendation
EMMA [97]	Multi-modal Annotation	W3C Recommendation
EmotionML [177]	Emotion Annotation	W3C Incubator Report
BML [106]	Behaviour Realisation	None - Draft proposition
FML [78]	Functional Behaviour Annotation	N/A
SemaineML [175]	Semaine API Formats	N/A

Table 5.4: The data standards for various conversational agents related functionalities and their W3C recommendation status

Whereas the standardisation of several data formats is an ongoing process, this approach seems to be more suitable for large scale web services, such as the Semantic Web, rather than using them for small scale, fast conversational agents. Google Protocol Buffers [67] presents a comparison between their current serialisation format and the XML parsing and conclude that their format is 10 to 100 times faster than xml.

MyBlock data encapsulation

Our approach defines another data representation level, which is Object-Oriented, allowing us to design object hierarchies corresponding to our main data types. The objects are extendible and independent from the data serialisation level. This allows us to change the serialisation level in the future, when either Google Protocol Buffers becomes a standard or W3C standards become stable and recommended.

MyBlock formalises only one root class, from which all the data transferred through the platform has to be implemented. This is similar to the Object Class hierarchy in most of the programming language. The second level of this hierarchy is represented by Identifiable Data, which defines a unique identifier for any encapsulated structure. This approach is useful in data synchronisation or to join data from different sources.

Any simple data exchanged inside the system, such as the text transcription received from the input is String Data. If a second level of annotation is added, the structure

is transformed into an Annotated data, which can attach multiple generic annotation levels to the data. For components dealing with affective information detection, a Valence Data specification has been created.

In order to prevent components from receiving or publishing invalid data, MyBlock has a "safety" mechanism, which checks all the data transferred between the components. This is done according to the data type policies defined on the component reception or publication channels.

All the data structures presented in Figure 5.4 have corresponding Java Objects implementations. This approach leads to a natural development of components, while the data formats remain open. Moreover, by using this approach, the memory footprint remains small due to the usage of primitives for any places where strings are not required. The usage of objects, rather than raw xml does not exclude the usage of web standards, such as BML, but it terms of optimality it is preferred to keep the memory footprint as low as possible.

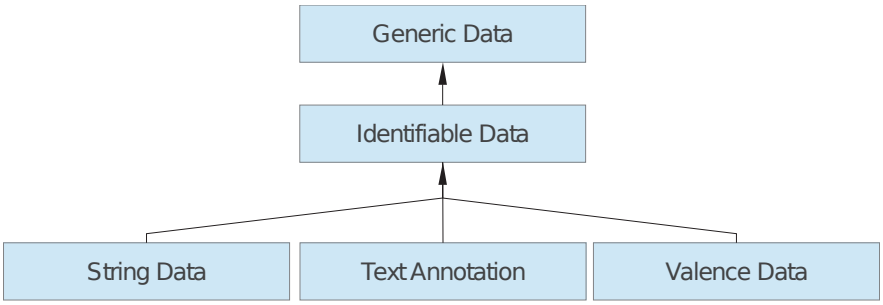


Figure 5.4: The simplified diagram of MyBlock data objects

5.3.5 Components, Services and Schedulers

In the distributed architecture presented so far, most of the components we described are just processing data. A second type of behaviour is to produce output, based on their internal states. This flow can be linked with a timeout, which triggers the desired behaviour. In order to provide these timeouts, synchronised to all the components, a special element was created: the schedule manager (or scheduler). Even if the component is not active, it has to subscribe to a schedule manager. The scheduler sends a heartbeat when a certain timeout is achieved, and that is broadcasted to every component subscribing to it. A component may choose to ignore this heartbeat. In MyBlock, the project may contain one or more schedulers, depending on the interpretation of the heartbeats.

The components encapsulate behaviour linked to a phase in the processing pipeline. A component processes a set of data types and output a list of results. MyBlock offers a series of abstract components, which, divided by their functionality, can be summarized by the following list:

- **Source**: represents a component which only publishes data to the system. System topics, such as heartbeat or system messages are not considered to be represen-

tative to the functionality of a component, so, even if a component receives data from these feeds, it is still considered as a source.

- **Sink**: describes the elements which only receive data. In theory, none of the system components can be a real sink, because every component is allowed to send system feedback. We consider to be sinks, all the components that only process data, without publishing any results.
- **Mixed Component**: represents a component that has the role of a Sink and the source, at the same time.
- **Living Component**: describes elements that react to the heartbeats published by the schedulers. This is the only functional difference between this element and a Mixed Component. In practice, for feedback generation, a Living Component could generate a message in case of a long inactivity

Each component can have assigned a set of properties, which acts as startup parameters. These are given at configuration time. In order to increase the readability of the source code and to add a second level of functional description to all the components, a special annotation has been created which describes the internal topics needed to configure a component and the properties used by this element. Listing 5.1 presents an example of this annotation with parameters. This annotation gives a new level of information about the component, but also provides a dynamic to reflect the component settings for instantiation.

```
@ConfigureParams(configureParams = "component.param",
                 outputChannels  = "component.data")
```

Listing 5.1: The annotation example for a component defines the configure parameters and the output channels

A service, in MyBlock, acts as a knowledge manager. The two sides of this design, with a client and a service, is flexible enough to be integrated in any processing pipeline. The client, in order to increase the speed, offers a priority cache ordered by the timestamp of the request. The whole generic behaviour of the service is encapsulated into a single entity, the AbstractService class, which deals with all the request and low level communication. A new service has to implement just a couple of new methods dealing only with the functional process, rather than querying over raw data.

So far, only two system services are defined into MyBlocks, and one experimental knowledge manager.

- **Computer Name Service**: It deals with all the computer name resolving issues. Since there are multiple ways to define a host or a physical device, we unified everything into a single service. The machine names are the standard physical identifier over MyBlock, and their IP or hostnames are resolved by this service. The actual calls to this service pass only once, when the connection between two components is made

- **Topic Service**: The topics in MyBlock have two forms: one is the raw, string format, which is used at the configuration time and it is human-readable, and the second, which is encoded into a numeric identifier, making it short and easier to work with. The Topic Service resolves this encoding and maintains a database of all the encodings at the running time.

The experimental knowledge manager offers simple synonym and definition relations for certain words. It is still under construction and at the end, it would have the role of a knowledge management database.

5.3.6 Example

In order to highlight the capabilities of the system, we compiled a simple example, one Source component and one Sink, with a single message, broadcasted at every heartbeat received from the Scheduler. The scheduler is configured to publish a heartbeat every 1000 milliseconds, which triggers the output of the publisher. The sink uses the Debug API and component, and publishes message on the debug topic.

Listing 5.2 presents the Java code for the Source component, which publishes a String Data on the *"test1"* internal topic (line 10). The *act* method is called every time a heartbeat is receive, and the *Test1* component publishes the *"Hello-t1: i"* message.

```
public class Test1 extends SourceComponent {
    private int i;
    public Test1(String port, ComponentConfig config) {
        super(port, config);
    }
    protected void setupComponent(ComponentConfig config) {
        i = 0;
    }
    public void definePublishedData() {
        addOutboundTypeChecker("test1", StringData.class);
    }
    public boolean act() {
        publishData("test1", new StringData("Hello-t1:"+(i++)));
        return true;
    }
}
```

Listing 5.2: Test1.java: The code for the functionality of the Source Component

Listing 5.3 presents the Java code for the Sink component, which receives a *StringData* (line 11). The *act* method does not have any functionality in this scenario and it returns a *false*, because the *act* does not execute any action. When *StringData* is received, the *handleData* method is called, which uses the Debug API to publish an inform to the Debug Component.

The listing 5.4 describes a simple MyBlock project, which runs the current Source and Sink.

1. For each machine configured on the system, at least one profile has to be created. In this case, the chosen *hostnames* are *machine1* and *machine2*. The *profile* names

```
public class Test2 extends SinkComponent {
    public Test2(String port, ComponentConfig config) {
        super(port, config);
    }
    protected void setupComponent(ComponentConfig config) {
    }
    public boolean act() {
        return false;
    }
    public void defineReceivedData() {
        addInboundTypeChecker(StringData.class);
    }
    public void handleData(GenericData data) {
        Logger.log(this, Logger.INFORM, data.toString());
    }
}
```

Listing 5.3: Test2.java: The functionality for the Sink Component

are *profile1* and *profile2*. For example, the *machine1* name is resolved as *host1*, by the Computer Name Service (Listing 5.5)

2. A *Scheduler* is defined on the port *1222*. This sends a heartbeat every 1000 milliseconds.

3. The two system services are declared: *CNService* (Computer Name Service) and *TopicService*. Each one has a different port assigned and a configuration file, if necessary. The Listing 5.5 shows how this configuration is made for the *CNService*.

4. In order to have a proper connection to the configured services, two corresponding clients are configured by assigning a connection string for each service: a machine name and a port, for each service

5. *LogComponent* is the one responsible for the debug API. As it is a regular component, it has a port assigned, a scheduler, and two subscribe channels for the existing components

6. *Test1* and *Test2* components are configured to communicate in a standard pipeline: Test1 publishes an external channel called "StringData.test1" and Test2 subscribes to this channel. As multiple components can publish channels with the same name, these are uniquely identified only by their full name: <channel name> @ <machine name>:<port>.

7. *Test1* component is deployed on *machine1*, under the *profile1*, whereas *Test2* uses *machine2*, *profile2*

The same components can be reused in other projects, as long they respect the same data formats. This approach provides a high code reusability and forces the creation of certain data formats, mandatory for proper exchange between components. Once these standards are created, any two compatible components can be coupled, decoupled or moved to another machine.

```
<project>
  <profile name="profile1" hostname="machine1">
    <scheduler>
      <port>1222</port>
      <timeout>1000</timeout>
    </scheduler>
    <services>
      <service name="org.ib.service.cns.CNService">
        <port>1221</port>
        <config>"cnsService.xml"</config>
      </service>
      <service name="org.ib.service.topic.TopicService">
        <port>1220</port>
      </service>
    </services>
    <clients>
      <client name="org.ib.service.cns.CNClient">
        <host>127.0.0.1</host>
        <port>1221</port>
      </client>
      <client name="org.ib.service.topic.TopicClient">
        <host>"machine1"</host>
        <port>1220</port>
      </client>
    </clients>
    <components>
      <component name="org.ib.logger.LogComponent">
        <port>1233</port>
        <scheduler>"machine1:1222"</scheduler>
        <subscribe>"org.ib.bricks.Test1.debug@machine1:1234"</subscribe>
        <subscribe>"org.ib.bricks.Test2.debug@machine1:1235"</subscribe>
      </component>
      <component name="org.ib.bricks.Test1">
        <port>1234</port>
        <scheduler>"machine1:1222"</scheduler>
        <publish>"StringData.Test1@test1"</publish>
      </component>
    </components>
  </profile>
  <profile name="profile2" hostname="machine2">
      <component name="org.ib.bricks.Test2">
        <port>1235</port>
        <scheduler>"machine1:1222"</scheduler>
        <subscribe>"StringData.Test1@machine1:1234"</subscribe>
      </component>
    </components>
  </profile>
</project>
```

Listing 5.4: The configuration file for a simple MyBlock project

```
<dns>
    <machine>machine1 @ host1</machine>
    <machine>machine2 @ host2</machine>
</dns>
```

Listing 5.5: cnsService.xml: The configuration file for the Computer Name Service having defined two different machines

5.3.7 Performance

In a previous section, we benchmarked the performance of the middleware used in other major DIS. A second benchmark is conducted, in order to test the performance of the

platforms built over this infrastructure. The setup of this experiment is identical to the one used for the middleware benchmark. The machine used for the test is a I7 Intel machine, at 1.6 GHz per core and 3.9 Gb of RAM. The operating system used for the test is on Ubuntu 12.04 Linux, with Oracle Java 1.7.

Table 5.5 presents the running times, in milliseconds, for SEMAINE and MyBlock. Table 5.6 shows the message throughput. The setup for these times is to send a series of random messages, for a given size, from one component to another. We choose to send random sequences in order to prevent the caching speed of ActiveMQ, the underlying middleware for SEMAINE Project, which applies a set of heuristics when the same message is sent over the network. Moreover, in order to prevent local traffic peaks, we sent 100 messages and presented only the average time.

System	**Message Size**					
	10	100	1,000	10,000	100,000	1,000,000
Semaine	0.38	0.38	0.36	0.62	3.35	24.72
MyBlock (ASF)[1]	0.53	0.51	0.54	0.68	2.85	19.28
MyBlock (Simple)[2]	0.33	0.31	0.31	0.55	2.88	18.79

[1]Automatic System Feedback (ASF) sends another message to inform the platform that the previous message has been processed successfully

[2]In this scenario the Automatic System Feedback has been disabled

Table 5.5: A running time comparison between SEMAINE and MyBlock. The time presented is expressed in milliseconds and the message size represents the length of the sequence sent over the platform

System	**Message Size**					
	10	100	1,000	10,000	100,000	1,000,000
Semaine	2,608	2,649	2,747	1,625	298	40
MyBlock (ASF)[1]	1,872	1,977	1,866	1,472	351	51
MyBlock (Simple)[2]	3,064	3,275	3,178	1,834	347	53

[1]Automatic System Feedback (ASF) sends another message to inform the platform that the previous message has been processed successfully

[2]In this scenario the Automatic System Feedback has been disabled

Table 5.6: A message throughput comparison between SEMAINE and MyBlock. The throughput presented is expressed in number of message per second and the message size represents the length of the sequence sent over the platform

For the message throughput, we represented the number of messages that pass between the two components in one second. Figure 5.5 represents this measure, side-by-side, on a logarithmic scale.

MyBlock is presented in two versions: with Automatic System Feedback (ASF) and non-ASF. This mechanism enables MyBlock to send a feedback message each time an action is successfully executed. This is executed in the case of sending and receiving

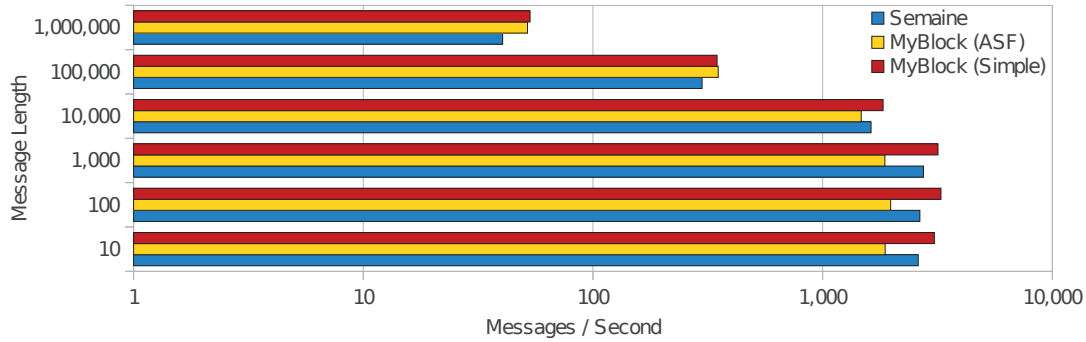


Figure 5.5: The performance comparison for SEMAINE and MyBlock, for message throughput representation. The representation is done on a logarithmic scale

a message. SEMAINE does not provide a similar mechanism, therefore, in order to achieve a fair comparison of the two systems, this feedback has been disabled.

The conclusion of this experiment is that currently MyBlock is faster than SEMAINE, when having the ASF disabled. For messages longer than 10,000 characters, the ASF does not increase the sending time. Since MyBlock targets a large spectrum of data types, both scenarios can be used in practical situations. The choice of a platform depends on the application. To achieve the best speed while sending data, MyBlock simple (non-ASF) is a good choice. To guarantee that a message has been successfully processed before sending the next one, MyBlock ASF is currently the only choice. In conclusion, the two versions of MyBlock are better that the current implementation of SEMAINE and the choice of a version depends on the scenario.

The MyBlock platform is just a component based architecture that allows fast and reliable data exchange between its elements. In order to transform this into a Distributed Interactive System, several elements with functionalities specific to interactive systems need to be constructed. Several functions (*Natural Language Understanding*, *Dialogue Management* and *Affective Feedback*, presented in Figure 5.1) can be implemented using a unified language, based on our extension over regular expressions: Syn!bad .

5.4 Syn!bad

Syn!bad is an extended regular expression language, for usage mainly in Natural Language Processing (NLP) applications. It uses the some extended POSIX Regular Expression [7] structures among others, more specific to NLP domain. As presented in the previous chapters, the learning phase for the detection algorithms require a feature extraction method. The knowledge extraction methods, involved in the Natural Language Understanding Process, use similar techniques to detect key concepts to be used in the Dialogue Management process. We propose this language to simplify the construction of these patterns.

The name, Syn!bad (also written: Synnbad, with double nn, instead of n!) is an acronym of *Synonyms [are] not bad*. This suggests that the main concepts of Syn!bad

are centred among synonyms processing, using different dictionaries.

Synonyms are independent structures, grouped in different sets, by their meaning. The most common grouping currently known is the WordNet synsets [123], which consist in grouping different words according to their semantics and part of speech. Each synset has a unique id, which permits an easy retrieval.

Syn!bad is embedded into a MyBlock component and is also available as a component for the AgentSlang platform. Nevertheless Syn!bad it is an independent language which can be implemented and distributed on its own. We present the language in the scope of basic knowledge extraction for IS, but this library can be extended to document classification, summarisation, topic extraction, etc.

In dialogue management, knowledge extraction or affect detection, having a set of patterns to extract the information simplifies the complexity of any system. Moreover, it gives a tool set flexible enough to process any data. Appendix A provides a formal view over the language, by presenting the BNF Grammar definition of Syn!bad .

5.4.1 Context

In IS, the knowledge extraction process is usually slowed down by the complexity of the rules describing a certain concept. Using regular expressions is an alternative, but in certain situations, composing rules for all the cases is impossible. Another approach is to group certain structures while making them more generic. For instance, instead of using a regular expression for matching the following sentence: `Ovidiu do you have water`, one could use `<name> do you <verb> <object>`. By using regular expressions, the variable structures are already supported by certain implementations.

The problem becomes more difficult when adding restrictions to the matched variables, especially in the case of `<verb>` and `<object>`. To our knowledge, the syntax of matching only variable structures while having a certain part of speech is not supported by any regular expression implementation.

A more complex situation is given by placing a synonymic relation restriction on the matched item. In our previous example, we would like to extract only the objects being synonyms of the word *water*. The synonyms usually introduce a certain fuzziness into a decision, since not all the meanings of a polysemantic word match the context of a given pattern. In this scenario, a certain restriction can be modelled, by adding a part of speech restriction on the word. For example, the word *good* has multiple meanings, such as *satisfactory in quality*, when employed as an adjective or *possession (object)*, when used as a noun. When matching a synonym of *good* with our rule, we can restrict this to only *nouns*, in which case the word *well*, as an adjective, is not matched.

5.4.2 Example

Given the previous context, we propose a first example of a Syn!bad pattern.

Based on the rules described above, we compile the following Syn!bad pattern, which also contains most of the features of the language:

```
$name <#*>? do you <VB*>* [some|RB*] [water#object]
```

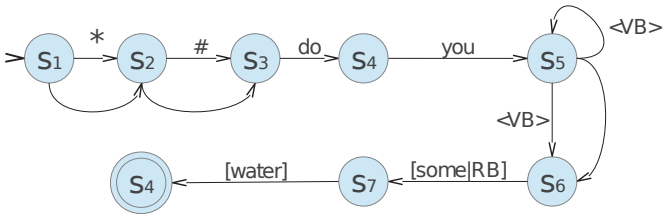


Figure 5.6: A Syn!bad example, presented as an automaton

- `$name` item represents a context free variable, which matches any single word and retrieve it as the *name* variable.
- `<#*>?` is an optional token that can match any punctuation mark. Moreover, the *#** represents a generic part of speech group matching punctuation marks.
- *do* and *you* are precise words matched by this expression.
- `<VB*>*` is an none-or-many token matcher, which restricts the element to match only a selected part of speech, in this case a verb.
- `[some|RB*]` represents a matcher for a synonym of the word *some*. Moreover, a restriction over the part of speech is added, which matches only adverbs.
- `[water#object]` this token is similar to the previous one, but it matches a synonym of the word *water* and recover the value of this word into the *object* variable.

In order to describe the whole matching process, the following sentence is given: `Ovidiu , do you want any aqua`

The result of the matching is: $\$name \leftarrow Ovidiu$ and $\#object \leftarrow aqua$, while `<#*>?` matches the comma mark (,), `<VB*>*` matches the single verb *want* and any is matched by the token `[any|RB*]`.

5.4.3 Implementation

The Syn!bad language has two levels, one is related to the grammar model, presented in the Appendix A. The second level concerns the implementation of this language, as an extension to the current capabilities of our knowledge extraction platform.

The patterns are compiled into an Deterministic Finite Automaton (DFA), completely written from scratch in Java language. We choose this representation since the DFA offers superior matching speed since the decision is mainly linear. Cox [43] presented a series of experimental arguments to sustain this decision.

The Deterministic Finite Automaton (DFA) is a type of automaton, where each state, for a given input, has at most one new state leading from the previous one. This makes the navigation through the states easier, since the possibility of exploration is always reduced to only one state or none. The finite status is given when our machine reaches one of the terminal states and the finish condition is fulfilled.

The simple token matchers use a simple word equality operator, whereas more complex items such as part of speech matchers and synonyms require more complex functions.

Concerning the part of speech restrictions, we propose two different types of labels. One is more strict, as recommended by the Penn Part-Of-Speech Tag System [170], which contains 45 different labels. The second is a functional grouping of the first system, called Generic POS, and contains only 5 labels:

1. `#*` groups all the punctuation marks into one single category: `$ # . , : ( ) " '`
2. `VB*` groups all the verb tags: `VB, VBD, VBG, VBN, VBP, VBZ`
3. `RB*` groups all the adverb tags: `RB, RBR, RBS`
4. `NN*` groups all the noun tags: `NN, NNS, NNP, NNPS`
5. `JJ*` groups all the adjective tags: `JJ, JJR, JJS`

The synonyms are currently extracted from the WordNet dictionary [123]. We use the synset identifiers, provided by WordNet, restricted by Part of Speech, when necessary. WordNet provides an index already split by part-of-speech, which makes the restriction conditions much easier to fulfil.

All the part-of-speech restrictions, synonyms and variable names are stores as a matching token, making possible to model our automaton as a DFA. All the tokens of a pattern are stored as a linked multi-list.

For each state, we assign a priority to each token, which makes the matcher decision even more simple. The top priority is assigned to the *optional token*, just before the *mandatory element*. This is done because it is more important to match an optional item, when possible, rather than a mandatory one. The process cannot continue without matching all the mandatory elements, therefore since the optional item can be skipped easily, it is important to match them before the mandatory items. The last priority is assigned to a consumer item, which is either a *skip* item or a *global variable* (a structure labelled `$name`). A *skip* is an element with the lowest priority assigned, which matches everything and it is used to define matching spaces. The current implementation uses a *skip* of 2 items defined by default.

Once the patterns are compiled, each one of them has an identifier assigned. These are not mandatory to be unique, and in certain situations can be useful to have duplicate identifiers, such as having polysemic expressions. For instance, the patterns: (`hello`) and (`hi there`), can have the same id (`id=greeting`), since both represent different forms of greetings.

When a pattern is matched, its identifier is returned, along with all the variables matched. The variables could be global: defined as `$name`, or local: `#name` which are defined by the part-of-speech or synonym matching tokens. For instance, the pattern (`hello $name`) matches the first word that comes after `hello` and stores in the `$name` variable, whereas the pattern `hello <NN*#name>*` matches the first *noun* that follows

the word `hello` and stores it in the variable `#name`. In fact, the `$name` variable is matching any word or punctuation mark, whereas `#name` variables stores the content matched by a specific token: part-of-speech or synonym.

Patterns, among the variable retrieval feature, have another level of static labels, name styles. A style is represented by a collection of pairs (label, value) assigned to each pattern. The functional value of this feature is represented by the possibility to assign a second level of annotation to a certain matcher. The label space is defined on the whole matcher container (all the pattern matchers added on the same list), and the label space is sparse, as well. When a matcher does not have a label defined, an '*' is automatically assigned to any undefined value.

To introduce the *styles*, we present a short example of this functionality. Table 5.7 defines three different patterns, each one having different styles assigned. Styles are comma separated, defined as a label=value pair.

Pattern	**ID**	**Style**
what do you want ?	p1	relation=familiar, rudeness=high
what can i do to help you ?	p2	rudeness=low
if i may ask , how could i help you ?	p3	relation=polite

Table 5.7: Syn!bad pattern examples, using the style definition features

The pattern *p1* has two values assigned to the styles relation=familiar and rudeness=high, *p2* defines a value just for rudeness=low therefore the relation becomes *, *p3* has the *polite* value assigned to the relation. Table 5.8 summarises these results.

Style	**ID**		
	p1	p2	p3
relation	familiar	*	polite
rudeness	high	low	*

Table 5.8: The values assigned to each style according to the pattern definitions from Table 5.7

Using of styles is not mandatory, but offers another level of granularity for the knowledge extraction model. The styles offer a complementary function for variable extraction and in case of large pattern databases, it also provides more information for the dialogue selection models and dialogue generation components.

Syn!bad is an extension to the POSIX regular expression language that employs special elements useful for NLP applications. These elements are synonymic or part-of-speech expressions that can be combined with regular word items. The pattern can be grouped by their semantic function and have various styles assigned, which makes the matching process useful for knowledge extraction and dialogue management. In fact, this language is a critical part of the AgentSlang platform, ensuring the Natural Language Understanding function of the system.

5.5 AgentSlang

AgentSlang is a collection of components, created on top of the MyBlock platform, which enables to build rich, distributed and fast IS. All the principles enumerated before for the MyBlock platform are therefore valid for AgentSlang.

AgentSlang provides a collection of 12 stable components and 1 experimental. All of them are presented in the following subsections, grouped by types and category. We present the components, both from the technical perspective, with informations for the internal channels, and component parameters. Moreover, every element, has a functional description among with the role/usage in the AgentSlang platform, as proof of concept for an interaction platform.

5.5.1 System components

System components are basic elements managed by the MyBlock platform, without being specific to AgentSlang. They are described in this section only to preserve a uniform presentation for all the elements of the platform.

Debug/Log Component

`Component: org.ib.logger.LogComponent`

The important aspect of the debug component is that the logging mechanism is entirely distributed and independent. In comparison with similar platforms, we provide exactly the same API for all the components. The logger manages the reception of debug messages in a centralised way. The current logging component provides only three levels of debug: critical, debug, inform. All the messages received are currently redirected to the console, but they can be forwarded to any logging library.

In case of a Logger not being configured into the system, the Log API prints any critical error on the console, in order to avoid missing system errors.

System Monitor Component

```
Component: org.ib.component.SystemMonitorComponent
   Channels: system.monitor.data
```

The system monitor receives status messages from all subscribing components. These status messages are re-broadcast to its subscribers, providing the source feedback. It acts like a proxy, by filtering any unneeded feedback.

Any component that would like to receive this feedback has to subscribe to the `system.monitor.data` topic of the SystemMonitorComponent.

Component Monitor

`Component: org.ib.gui.monitor.MonitorComponent`

The monitor component provides a complex set of functionalities for system monitoring. Similar to the LogComponent, the monitor receives all the debug messages from all the subscribed components. An interesting function is provided by the level and source filters for debug messages, which makes the debug and monitoring process easier. Figure 5.7 shows an example of a system configuration, monitored with the Component Monitor.

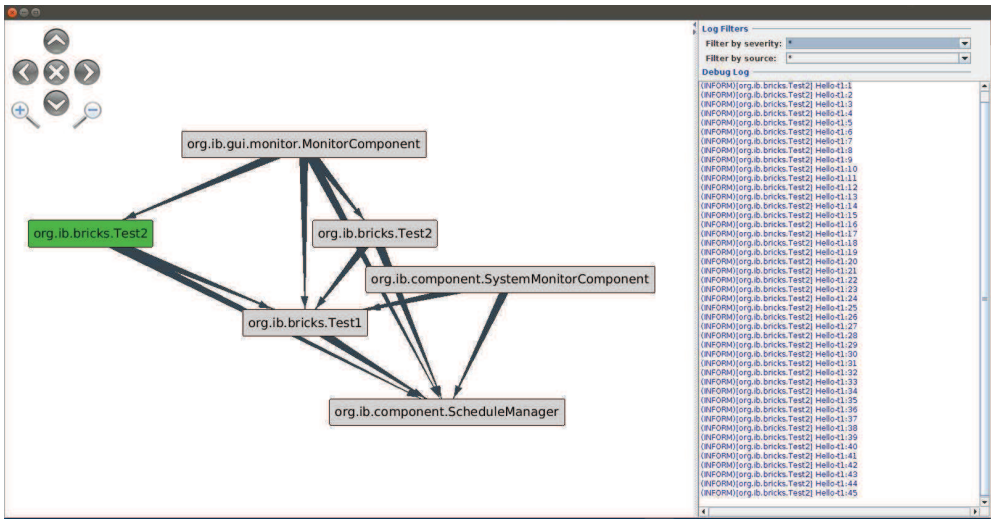


Figure 5.7: The Component Monitor displays the debug log, filtered by a selected level, and the component activation (in this case, the green component)

A second function is the component interaction graph, with a message activation highlight. This component is rendered using GraphStream [152], which is a fast graph rendering library, used for dynamic graphs.

The component can also subscribe to the component feedback channels, similar to SystemMonitorComponent, and it highlights the component usage, each time a message is processed by that element.

5.5.2 Input and Output components

Text Component

```
Component: org.agent.slang.inout.TextComponent
   Channels: text.data
```

The Text Component acts both as a Source and a Sink for text data types. It can send StringData to the system. Moreover, it can subscribe to any channel and displays the received message as plain text. Figure 5.8 shows the main window of this component.

Voice Proxy Component

```
Component: org.agent.slang.in.VoiceProxyComponent
   Properties:
      voiceProxy
```

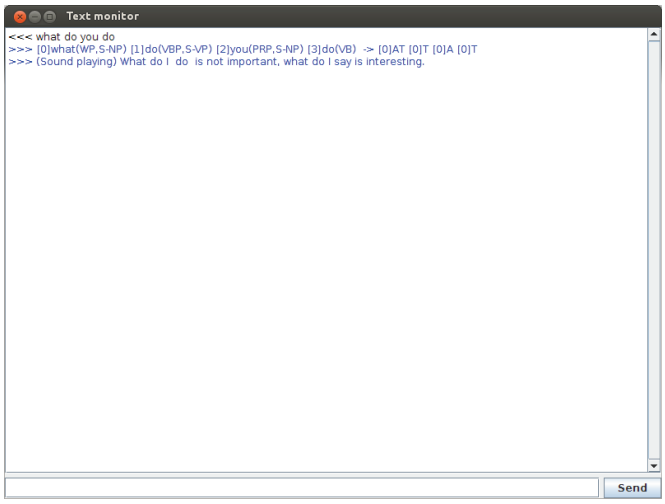


Figure 5.8: The Text Component has the ability to: to subscribe to multiple channels, display all the data received and send text input

```
    voiceBTuuid
    voiceBTmac
Channels: voice.data
```

Accurate real-time transcription, is very challenging. A few commercial options exist, providing a high accuracy for domain specific transcriptions. For instance, Nuance Dragon Speech Recognition Software [133] provides a high accuracy for medical and legal domain. Moreover, most of the modern operating systems offer accessibility support for disabled persons, and integrate good automatic speech recognition software.

The Open Source projects, such as HTK [208], Sphinx 4 [109], Mobile Sphinx [84] or Julius [111] are very promising from the state-of-art perspective, but the acoustic models currently embedded in these project are very basic, without any use in real dialogue applications. VoxForge [201] is trying, since 2005, to fill the gap between industrial models and open source, but the accuracy of the results remains less than 60 %, for clear recorded voice. For radio quality or recordings with regular microphones, the recognition rate is much less.

On mobile platforms, the situation seems encouraging. Due to the recent success of the Siri [10], Google Speech API [68] and their integration with iOS and Android, other companies decided to provide full and cheap support on their API for mobile devices. Unfortunately, this kind of access is restricted to mobile platforms.

Based on this accuracy obtained by mobile platforms and the support for multiple languages, we decided to integrate an Android application into our system, which would act as a smart "microphone", by forwarding the transcription to our system. The Voice Proxy Component does exactly this: it acts as an entry point in the system for the transcription provided by the Android application. All the content is forwarded to the *voice.data* channel and encapsulated as a StringData type.

The proxy can be configured in two modes, depending on the access type and device support:

1. Socket mode, which starts a TCP server on the port given by the `voiceProxy` property. This is used by the Android client application to connect. This is useful in case both the Android device and the AgentSlang system have access to the same network

2. Bluetooth mode, which allows the Android device to be connected as Bluetooth pair with a computer. The `voiceBTmac` property provides an optional selection by MAC address of the Bluetooth device and `voiceBTuuid` describes the Universally unique identifier (UUID) [143] used by the bluetooth protocol to engage connection. An UUID is a unique descriptor, standardized by the Open Software Foundation (OSF) as part of the Distributed Computing Environment (DCE), and is similar to the TCP host-port pair. It provides access to certain services, in this case bluetooth.

There is currently no restriction on the number of voice proxies to be configured in the system, but only one Android device can be configured for each Voice Proxy Component.

MaryTTS Component

```
Component: org.agent.slang.out.marytts.MaryComponent
   Properties: locale
   Channels: voice.data
```

MaryTTS [178] is one of the very popular toolkits for affective speech generation. It provides support for German, British and American English, Telugu, Turkish, Russian and Italian languages, maintained by DFKI Laboratory (Germany) and an experimental support for French, provided by Speech Research Group at ParisTech (France). This components has been part of the SEMAINE Project [175].

Our component provides a basic integration, with StringData or AnnotatedData as input, and forwarding the speech generated content to the *voice.data* internal channel.

iSpeechTTS Component

```
Component: org.agent.slang.out.ispeech.ISpeechTTSComponent
   Properties:
      voice
      apiKey
   Channels: voice.data
```

iSpeech TTS [90] is a technology developed by the iSpeech Corporation, and provides a natural voice support for a large list of languages. The support is provided on a Software as a Service (SaaS) basis, with a flexible pricing service. It also offers support for both desktop and mobile platforms.

Our component integrates basic support for the text-to-speech API, providing configuration for two parameters: the *voice*, which allows the selection of the style of the

voice, such as male or female and language, and the *apikey*, which is the access key to the service.

This component demonstrates that commercial components can be integrated into our system.

5.5.3 Natural Language Processing and Dialogue Models/Components

Senna Annotator

```
Component: org.agent.slang.annotation.SennaComponent
   Properties:
      sennaParams
      sennaPath
   Channels: senna.data
```

Most of the processing components used in an IS use part-of-speech tagging, segmentation or chunking. The part-of-speech tagging supposes a multi-label annotation task, with labels based on the Penn Tag Set Guideline [170] for English . It is one of the classical challenges for Natural Language Processing and several toolkits have implementations in almost every programming language.

The segmentation task (chunking), which is also an annotation task, involves splitting the sentence into syntactically correlated parts of words. It has been originally proposed by Abney in 1991 [2]. Since then, the problem has been proposed into several challenges, including the latest CoNLL-2000 task [194], which remains the benchmark for this issue.

The Name Entity Recognition task involves the detection of word structures which may be points of interest, such as places, names or organisations. This task is highly linked to the context and sometimes subjective. The latest benchmark for this task is proposed by CoNLL-2003 task[195], which describes the problem from the language independence aspect.

SENNA [39] proposes a solution to all these Natural Language Processing tasks. The system obtains state of art precision for English language[5], or better, by training a large scale neural network, by preserving the model as problem independent. Moreover, SENNA offers superior speeds compared to similar systems.

Our current version of AgentSlang is written in Java and SENNA is coded in standard C, so we decided to integrate the toolkit as an external execution point. This involves the configuration of the *sennaPath*, which points to the exact location of the executable. Moreover, SENNA can be configured to solve a certain set of tasks, from all the ones proposed: chunking, part-of-speech tagging or named entity recognition. The *sennaParams* configures the set of tasks to be executed. The results are published on the *senna.data* feed as an AnnotatedData, where each level corresponds to the selected SENNA task.

[5]The Tag Set used for annotation is the one recommended by the CoNLL competition task

Metaphone Encoding Component

```
Component: org.agent.slang.annotation.MetaphoneEncodingComponent
   Channels: metaphone.data
```

In dialogue and web semantics, the accuracy of the results can be sometimes degraded by the precision of the transcription. A simple solution to this problem is to encode all the words from the original query into a phonetic form, rather than using the original words. This is linked to the fact that strong phonetic languages, such as English, need a good pronunciation or spelling to obtain very accurate results. In particular, the precision of the transcription decreases dramatically for non-native speakers.

The Metaphone encoding [148] is a proposition to solve this problem. It has been proposed by Lawrence Philips in 1990, as an improvement of the original Soundex algorithm [27]. A second version of this algorithm was created, called DoubleMetaphone [149], which included encoding rules for non-english words familiar to americans, as well. Currently, there exists a third commercial-version of this algorithm, Metaphone 3 [147], which covers 99 % of all the words used in American English. The principle of all these algorithms is to construct a set hand-crafted rules to encode a word according to the phonetic rules of American English, which creates a high similarity for words with close pronunciations.

In MyBlock, we use the DoubleMetaphone version of the algorithm, which takes a string as an input, and provides the phonetic encoding to the *metaphone.data* channel. This strategy is efficient only for English, since the algorithm was created to cover the rules of this language.

Template Extractor Component

```
Component: org.agent.slang.dm.template.TemplateExtractor
   Properties:
      dialogueModel
      commandModel
   Channels:
      templateExc.dialogue.data
      templateExc.command.data
```

The Template Extractor Component is based on the Syn!bad language and is used to collect all useful information from a sentence. The current model is hand-crafted and has 10 patterns for presenting the agent, the description of capabilities and introducing the speaker.

The patterns are grouped into two different models: the dialogue and the command. The only difference is the semantics of the patterns. For commands, the pattern matcher is configured in a very strict mode, where no synonyms are explored, and the sentence should start with the first word of the pattern. This decision was made due to the semantic strictness of a command and to avoid any ambiguity in the interpretation

process. All the template identifiers extracted using the command model are sent via the *templateExc.command.data* channel.

The dialogue model allows the usage of synonyms and partial matchings. Moreover, the matching process is not mandatory to start from the beginning of the sentence. The template identifiers resulting from the matching process are send to the *templateExc.dialogue.data* channel.

Command and Dialogue Interpretor

```
Component: org.agent.slang.dm.template.CommandInterpreter
   Properties: commandModel
   Channels: command.data

Component: org.agent.slang.dm.template.DialogueInterpreter
   Properties: dialogueModel
   Channels: dialogue.data
```

The two components for command and dialogue interpretation use the same reactive model for behaviour generation. The current implementation maps directly a reply action to a pattern identifier, extracted with the TemplateExtractor. The reply action represents a sentence or a command (in case of the Command Interpretor) that is interpreted afterwards. The reply action patterns allow variable substitution, similar to the Syn!bad language:

```
item1 item2 $variable1 item3 $variable2 item4
```

where `item1,item2,item3,item4` are actual words used to generate the reply and `$variable1,$variable2` are two variables. In the current implementation, since the reply action is mapped directly to the pattern identifier, the variable names are mapped as well to the variables extracted previously. If no such variable can be extracted, the process fails and no pattern is generated.

The Dialogue Interpreter needs a *dialogue model* which describes the generation patterns and builds a reply action which is transmitted to the *dialogue.data* channel. The Command Interpretor does the same, but it can execute or generate a system command (i.e. turn on/off a switch, play a song).

5.5.4 Affective Feedback

Valence Extraction Component

```
Component: org.agent.slang.feedback.ValenceExtractorComponent
   Channels: valence.data
```

The Valence Extraction Component is a generic template for any component that extracts valence, from any type of data. Our approach is an affect oriented system, where multiple Valence Extractors fusion their data. We believe that the feedback

offered by these components could improve the quality of the whole system, as concluded in the **Chapter 2** and **3**.

The architecture of this component is simple and intuitive, so the implementation of any valence extraction should be straightforward. As a case study, we present two different approaches to implement such an extractor.

1. A multi-modal valence extractor, build on top of a fusion model

2. A pattern based model, build on top of the affective contextonyms

The model for the multi-modal valence extractor has been described previously, in **Chapter 2** (Section 2.7). Assuming the usage of an NGram + Smile fusion model, this algorithm takes as an input a sentence annotated with the part of speech of each word and a Smile intensity, extracted using a Smile extraction algorithm (i.e. OKAO[6]). The sentence is used to extract the NGram features, while the Smile intensity is converted to a Smile level, according to our strategy presented in **Chapter 2** (Subsection 2.7.4).

The pattern based model, using the affective contextonyms, is more complex and requires using multiple techniques described along this thesis. First, the affective contextonyms model is presented in **Chapter 3** (Section 3.4). It consists in a context graph model for affective words. Since the usage of this graph is not as simple as other linguistic resources and requires multiple queries in graph structured data, we can compile it as a series of Syn!bad patterns, with a different style label associated to each word. Moreover, for some contextonyms, a synonymic collapse can be allowed:

1. In case of direct WordNet [123] synsets, where the synonyms are not found as part of another contextonym

2. Two contextonyms can be collapsed if the valences of each synonym words is the same and no words without correspondence can be found in any of the two contextonyms

In this case, a contextonym consisting in the words (*cyclone, tropical* and *island*) generates the following Syn!bad pattern: `[cyclone] tropical island`, with the following style associated: `w1=-0.125,w2=-0.25,w3=-0.25`. A model of such patterns can be compiled, making the whole pattern matching process easier by using Syn!bad language.

Summary

Table 5.9 compiles a list of all the available AgentSlang components, grouped by their function in the system architecture.

[6] `http://www.omron.com/r_d/coretech/vision/okao.html`

Component	Function	Type
Log Component	Debug/Logging	Sink
System Monitor Component	Event Monitoring	Mixed
Component Monitor	Platform Monitoring	Sink
Text Component	Text Input/Output	Mixed
Voice Proxy Component	Speech Input	Source
MaryTTS Component	Speech Synthesis	Sink
iSpeechTTS Component	Speech Synthesis	Sink
Senna Annotator	POS Tagging, Chunking, NER	Mixed
Metaphone Encoding Component	Phonetic Encoding	Mixed
Template Extractor	Knowledge Extraction	Mixed
Dialogue Interpreter	Dialogue Generation	Mixed
Command Interpreter	Dialogue and Command Generation	Mixed
Valence Extractor Component	Valence Extraction	Mixed

Table 5.9: A summary of all existing AgentSlang components

5.6 Discussion

Table 5.10 presents a comparison of key features across main state of art systems and AgentSlang. The core functions of our approach are the Dialogue Management and Affect Oriented design, based on specific components that support this claim. Currently only AgentSlang, Semaine and Companions projects use Affect Oriented features. For Dialogue Management, AgentSlang, Companions, Mirage and GECA offers it. VHMsg is more a middleware library for building interactive agents, with no specific system implementations. On the system management aspect, currently only AgentSlang and Semaine provides tools for such a purpose, whereas the real-time system messages, that provide informations about the state and status of all the components, are supported only by AgentSlang.

On the technical level, several advantages of AgentSlang could be presented. Our platform uses MyBlock as a middleware, which is based on ZeroMQ. These platforms are actively supported and available on multiple operating systems: Android, Linux, Mac and Windows. In comparison, ActiveMQ is available only for Linux, Mac and Windows. Moreover, ZeroMQ is providing support for all major programming languages, which gives the opportunity to migrate MyBlock to other environments. Currently we use only the publish/subscribe and request/reply pattern over TCP sockets offered by the ZeroMQ. ZeroMQ also supports inter-process communication and shared queues, which could increase the communication speed for components found on the same machine.

Two of the major differences between MyBlock and other platforms are that we focus on data-oriented design rather than the source and by not using blackboards. We

consider that the data source is not important for Distributed Interactive System, since trust among components is guaranteed by the builder of the system. At most, components could provide a level of confidence for certain tasks. Moreover, the blackboard as a fundamental mechanism for knowledge sharing in AI is replaced by the services, modelled by MyBlock. A service offers similar functionality through the request/reply protocol, being more flexible and offering more robust data synchronisation mechanisms.

Finally, we have shown that the performance of ZeroMQ and MyBlock is better than ActiveMQ and Semaine, which are the platforms with the best performance in our study.

MyBlock and AgentSlang offer a clear separation of the platform and application level, according to their role. AgentSlang is a library that offers several component that can be easily implemented and transferred to other platforms. This offers the possibility to implement various algorithms, from the same category into a common platform. By doing so, one is able to benchmark the performance of various components and choose the most suitable for its purpose. Currently, components proposed include automatic speech recognition, speech synthesis, knowledge extraction, dialogue management and affective feedback extraction.

AgentSlang is a platform build around the idea of Open Source software for Distributed Interactive Systems that offers Affective Feedback Detection and Dialogue Management. Currently it is the only system offering all these features (Table 5.10).

	AgentSlang	**Semaine**	**Companions**	**Mirage**	**GECA**	**VHMsg**
Middleware Platform	MyBlock (ZeroMQ)	ActiveMQ	Inamode	Psyclone	OpenAir	ActiveMQ
Integration Approach	pub/sub	pub/sub	plain socket	blackboard	blackboard	pub/sub
Operating Systems	ALMW[1]	LMW[2]	Unknown	LMW[2]	Unknown	LMW[2]
Data Interface	custom Objects	multiple[3]	XML	string	XML	string
System Management	Yes	Yes	(No)	No	No	No
System Events	Yes	No	(No)	No	No	No
Actively Maintained	Yes[4]	Yes	Unknown	(No)	(Yes)	(Yes)
Platform Licence	GPL+[5]	LGPL	Proprietary	Unknown	Unknown	LGPL
Dialogue Management	Yes	No	Yes	(Yes)	Yes	No
Affect Oriented	Yes	Yes	Yes	No	No	No

[1]Android, Linux, Mac, Windows

[2]Linux, Mac, Windows

[3]String, XML, binary data converted to String

[4]Currently under development

[5]GPL+ means GPL and French CeCILL

Table 5.10: A comparison of key features of AgentSlang and existing State of Art Interactive Systems

Part III

Conclusion and Future Work

Conclusion

The issues covered by this thesis are linked to various Affective Computing and Interaction problems. The computer should be able to detect emotions and reply accordingly.

The Affective Computing part is represented by several Detection Algorithms: a text-based emotion classification method using Self Organizing Maps and a valence classifier based on multi-modal features and Support Vector Machines. Moreover, due to an observation made while developing the classification algorithms and since one representative dictionary for sentiment analysis (SentiWordNet) carries a large number of conflicts, we also proposed a method to solve these conflicts. From the Interaction perspective, this thesis approaches two issues: an experiment created for collecting rich interactive data, in a story telling environment and an architecture for a Distributed Interactive System.

Detection of User's Affective Feedback

The field of Emotion Detection focuses on two major aspects: creating better detection algorithms and building more accurate affective dictionaries. We approached both of these issues in this thesis.

Our first experiment is based on a Self Organizing Map classifier, which is easy to train, but very versatile for fuzzy classification. We used a feature extraction using a Latent Semantic Analysis on text, which served as support for our classifier. The approach has been validated on a well known corpus for semantic affect recognition: SemEval 2007, task 14. For this purpose, we managed to obtain a model that provides a good balance between precision and recall, for the given corpus.

This first approach uses only text to extract features. Several recent studies in Affective Computing propose multi-modal approaches. Our second experiment is conducted as a multi-modal classification study on a Youtube corpus. The smile, as a gesture feature, is fused with several other features extracted from textual data. For this purpose, we generate different feature configurations to study the smile influence. These features are used with a two level linear Support Vector Machine, which offers the possibility to study in more details the classification process. On the Youtube corpus, by using this approach, we managed to obtain the best results, compared to the original Morency et al. [126] approach, with a method that is fast enough for an interactive system.

Several issues regarding the classification precision and recall for Affective Computing are linked with the dictionaries used. These are either manually constructed with a size too small to cover all the semantic cases or very large in size but carrying a large number of internal conflicts. Decreasing the number of inconsistencies in a dictionary directly improves the precision of the method using it. We proposed to decrease the number of inconsistencies of an existing dictionary (SentiWordNet) by introducing context. The context is modelled as a contextonym graph, built using a subtitle database. We managed to obtain a low conflict rate, while the size of the dictionary is preserved. By using our method, our goal is to obtain a large contextualised affective dictionary

that can be used for emotion classification tasks.

The contextonym is modelled as a strong semantic relation between the terms, similar to synonyms. In fact, these are cliques in a graph of word co-occurrences. Since none of the existing algorithms could handle large and dynamic graph structures, the clique extraction algorithm used for this purpose was designed for building the contextonym graph. Our algorithm, the Dynamic Distributable Maximal Clique Algorithm (DDMCE), was successfully validated on various random generated databases. One of the strong points of this algorithms is the it addresses the issue of processing in a distributable way large and dynamic data simultaneously. Moreover, the most important validation is represented by the ability to successfully process the data needed to generate our contextonyms model.

Affective Interactive Systems

From the interaction perspective, in order to obtain a rich corpus for the problem of Child-Machine interaction, we created an innovative storytelling environment. From the psychological perspective, this experiment is a validation of the interactive engagement between a child and a virtual character. We measured the difference between a setup having the virtual character as a narrator or a psychologist in video conference mode, by using various communicative features. The only difference, we observed, between the two scenarios is in the communication modality. This experiment also lead to the development of a new Wizard of Oz platform (OAK), that allows online annotation of the data. This environment allows the design of various reactive dialogue models, which can be tested and integrated into our future system.

The final aspect of this thesis is the proposition of a new architecture for a Distributed Interactive System. By using a component based design approach, we model a component structure that is light and simple enough to allow the integration of any existing algorithm. We propose several components for knowledge extraction (Syn!bad), reactive dialogue management and affective feedback detection, among other classic components (i.e. Automatic Speech Recognition, Text to Speech). This platform intends to be the foundation for several affect detection algorithms, starting with all the algorithms previously presented.

Future work

From the affect detection perspective, the multi-modal approaches should be investigated further. Our experience with the ACAMODIA Project showed that in practice, no modality carries more importance than the others. In the storytelling experiment, the feedback is sometimes recovered from speech, gestures, postures, smiles or eye-gazing. In almost all the situations, the feedback is recovered only from one modality, while the other are missing or occluded. For example an occlusion phenomenon in speech appears when a noise covers the dialogue making the word recognition impossible, while an occlusion phenomenon in gestures appears for instance in a situation where the face

is covered by the user's hand. Working with children makes this problem more difficult because all these issues appear with a high frequency.

One of the current limitations of our valence detection algorithms is not to use valence shifters [155] during the experiments. One of the classic example of this category is by introduction the negation as part of the models. In practice, we taught about the negation in our multi-modal experiment, but it would not influence the final results because the presence of negated phrases was less than 1 % of the whole corpus. Another perspective for this problem is offered by the usage of "valence intensifiers", which similar to the valence shifters act as modifiers for certain words. Our initial hypothesis was that the smile would act as an intensifier for text content, similar to the "valence intensifiers'. This was not confirmed by our multi-modal experiment unfortunately.

Currently, from the machine learning perspective, two algorithms are used during this thesis. One is the Self-Organizing Maps and the second is the linear SVM. In the future, more complex models could be investigated, such as Neural Networks, multi-kernel SVM, SVM with an RBF kernel or CRF. The last model has been particularly successful in the modelling of linguistic resource and has been used in the NLP community to describe generative models.

From the dictionary conflict resolution, our current approach uses subtitles to compile a non-formal, dialogue style, linguistic model. In future, for different language styles a corresponding context graph could be constructed. This could improve the detection results for more formal environments. In the end, a large context structure can be compiled, with words clustered by style, domain and part-of-speech. Each of these propositions raises issues related to the Big Data Processing domain, which currently have not been discussed by the thesis.

Another idea that could be studied as a perspective is not to attach the valences to each word in a contextonym, but to the whole clique. This could lead to more stable results and less conflicts. Nevertheless, a different set of strategies for valence distribution need to be proposed for this task.

One of limitations of our current context model is that we did not conduct multiple experiments on various window size. This could build richer semantic relations between distant words of a phrase. In the end, this is a problem of window calibration. Moreover, in the same context of future investigations in the influence of certain parameters over our context model, a less aggressive filtering algorithm could be applied. This was done mainly for performance reasons so far, but with the current optimisations of the DDMCE algorithm, this could be redone.

Another aspect that has not been covered by this thesis is the investigation of alternative models for the context graphs. Currently the context is modelled as a "strong clique", which means that all the words that are part of this structure need to be linked with all the other words. From the linguistic perspective, in order to model this relation very accurately, you need a very large linguistic resource to cover all the context situations. Moreover, this problem could be explored through a "soft clique" perspective, where the words do not have links with all the other parts of this

structure. In fact, this idea would raise some theoretical issues related to graph theory. In the end, we believe that the problem itself is just a matter of calibrating the filtering parameters and finding linguistic resources large enough to model the context (such as movie subtitles).

The AgentSlang system offers a good foundation for building Affective Interactive Systems. Nevertheless, the number of usable components needs to increase. Currently, we propose at least one element to solve each representative problem in our design flow, but more need to be integrated. The dialogue management module, one of the critical parts of our proposition, is currently a reactive approach. The state based model, developed for the ACAMODIA Project needs to be integrated as well. Moreover, a more complex management based on Dialogue Games Theory [51], like the one we proposed for Ales et al. [6], could be developed.

Our final proposition, MyBlock has been validated only in terms of performance against a similar system, the SEMAINE platform. The technical validation is only the first step and a more extensive study of integration and acceptability needs to be conducted with human users. For this purpose, we propose two directions: *a*) a demonstrator having only Affective Feedback detection and synthesis capabilities, with a basic reactive dialogue management, similar to the Sensitive Artificial Listener proposed by SEMAINE *b*) a system having complex dialogue management based on ACAMODIA Project or Dialogue Games Theory [6, 51]. The first direction allows to compare our platform with SEMAINE, using the same scenario. Whereas, the second approach is more complex and integrates all the components currently developed (i.e. Affective Feedback Detection, Knowledge Extraction, Dialogue Management). This study can be conducted in a storytelling environment, similar to ACAMODIA, or a scenario dependent task, such as *"How was your day ?"* proposed by the Companions Project.

Part IV

Appendix

APPENDIX A

Syn!bad Syntax

The BNF Grammar of the Syn!bad syntax is defined as following:

⟨*expression*⟩ ::= ⟨*token*⟩ '␣' ⟨*expression*⟩ | ⟨*token*⟩

⟨*token*⟩ ::= ⟨*pattern*⟩
| ⟨*pattern*⟩ '*'
| ⟨*pattern*⟩ '?'
| ⟨*pattern*⟩ '{' ⟨*number*⟩ ',' ⟨*number*⟩ '}'

⟨*pattern*⟩ ::= ⟨*word*⟩
| '<' ⟨*POS_Structure*⟩ '>'
| '[' ⟨*synonym*⟩ ']'
| '$' ⟨*variable*⟩

⟨*POS_Structure*⟩ ::= ⟨*POS*⟩ ('#' ⟨*variable*⟩)?

⟨*POS*⟩ ::= ⟨*PennPOS*⟩
| ⟨*GenericPOS*⟩

⟨*synonym*⟩ ::= ⟨*word*⟩ ('|' ⟨*POS*⟩)? ('#' ⟨*variable*⟩)?

⟨*number*⟩ ::= [1-9] [0-9]*

⟨*word*⟩ ::= [a-z]+

⟨*variable*⟩ ::= [a-z0-9]+

⟨*PennPOS*⟩ ::= ‘JJ’ | ‘RB’ | ‘DT’ | ‘TO’ | ‘RP’ | ‘RBR’ | ‘RBS’ | ‘LS’
| ‘JJS’ | ‘JJR’ | ‘FW’ | ‘NN’ | ‘NNPS’ | ‘VBN’ | ‘VB’ | ‘VBP’
| ‘PDT’ | ‘WP$’ | ‘PRP’ | ‘MD’ | ‘SYM’ | ‘WDT’ | ‘VBZ’ | ‘‘’’
| ‘#’ | ‘WP’ | ‘’’’ | ‘IN’ | ‘$’ | ‘VBG’ | ‘EX’ | ‘POS’ | ‘(’
| ‘VBD’ | ‘)’ | ‘.’ | ‘,’ | ‘UH’ | ‘NNS’ | ‘CC’ | ‘CD’ | ‘NNP’
| ‘PP$’ | ‘:’ | ‘WRB’

⟨*GenericPOS*⟩ ::= ‘#*’ | ‘VB*’ | ‘RB*’ | ‘NN*’ | ‘JJ*’

APPENDIX B

DDMCE: Running times for Static Graphs

			Time			Cliques		
n	ρ	c	$\bar{x}$	σ^2	$\sigma^2/\bar{x}$	$\bar{x}$	σ^2	$\sigma^2/\bar{x}$
1,000	.01	1	.342	.032	.092	4,723.30	63.62	.013
	.01	2	.250	.004	.015	4,723.30	63.62	.013
	.01	4	.240	.014	.060	4,723.30	63.62	.013
	.03	1	.598	.011	.019	10,486.00	48.66	.005
	.03	2	.557	.019	.034	10,486.00	48.66	.005
	.03	4	.543	.027	.050	10,486.00	48.66	.005
	.05	1	.776	.014	.019	21,158.40	254.84	.012
	.05	2	.836	.065	.078	21,158.40	254.84	.012
	.05	4	.996	.069	.069	21,158.40	254.84	.012
	.07	1	.992	.015	.015	45,846.40	479.08	.010
	.07	2	.886	.100	.113	45,846.40	479.08	.010
	.07	4	1.071	.077	.072	45,846.40	479.08	.010
	.10	1	1.617	.029	.018	99,671.90	940.76	.009
	.10	2	1.215	.034	.028	99,671.90	940.76	.009
	.10	4	1.433	.164	.115	99,671.90	940.76	.009
	.15	1	4.683	.112	.024	349,312.90	5,613.65	.016
	.15	2	3.217	.313	.097	349,312.90	5,613.65	.016
	.15	4	3.266	.114	.035	349,312.90	5,613.65	.016
	.20	1	16.819	.432	.026	1,203,453.90	14,956.69	.012
	.20	2	9.527	.249	.026	1,203,453.90	14,956.69	.012
	.20	4	9.781	.251	.026	1,203,453.90	14,956.69	.012
	.25	1	67.404	5.376	.080	4,334,551.90	67,956.84	.016
	.25	2	37.192	1.131	.030	4,334,551.90	67,956.84	.016
	.25	4	35.882	.470	.013	4,334,551.90	67,956.84	.016

			Time			Cliques		
n	ρ	c	$\bar{x}$	σ^2	$\sigma^2/\bar{x}$	$\bar{x}$	σ^2	$\sigma^2/\bar{x}$
	.30	1	281.932	8.606	.031	15,935,818.00	359,968.05	.023
	.30	2	160.485	12.376	.077	15,935,818.00	359,968.05	.023
	.30	4	149.339	4.244	.028	15,935,818.00	359,968.05	.023
5,000	.01	1	1.687	.109	.064	96,514.20	252.58	.003
	.01	2	1.266	.041	.033	96,514.20	252.58	.003
	.01	4	1.324	.086	.065	96,514.20	252.58	.003
	.03	1	9.010	.195	.022	514,312.60	1,791.44	.003
	.03	2	5.673	.171	.030	514,312.60	1,791.44	.003
	.03	4	5.363	.122	.023	514,312.60	1,791.44	.003
	.05	1	30.805	1.731	.056	1,789,995.60	4,671.97	.003
	.05	2	18.585	.484	.026	1,789,995.60	4,671.97	.003
	.05	4	17.828	.142	.008	1,789,995.60	4,671.97	.003
	.07	1	90.080	.457	.005	4,081,340.80	10,908.97	.003
	.07	2	54.499	.654	.012	4,081,340.80	10,908.97	.003
	.07	4	52.718	.763	.014	4,081,340.80	10,908.97	.003
	.10	1	408.919	6.737	.016	18,442,189.70	98,336.50	.005
	.10	2	245.741	7.739	.031	18,442,189.70	98,336.50	.005
	.10	4	235.444	3.191	.014	18,442,189.70	98,336.50	.005
10,000	.01	1	13.114	.436	.033	349,267.60	393.07	.001
	.01	2	9.522	.539	.057	349,267.60	393.07	.001
	.01	4	9.246	.994	.108	349,267.60	393.07	.001
	.03	1	99.625	1.685	.017	3,738,091.30	7,663.35	.002
	.03	2	62.055	.973	.016	3,738,091.30	7,663.35	.002
	.03	4	57.786	.471	.008	3,738,091.30	7,663.35	.002
	.05	1	467.644	2.260	.005	12,177,490.30	24,768.75	.002
	.05	2	286.653	2.708	.009	12,177,490.30	24,768.75	.002
	.05	4	269.613	2.972	.011	12,177,490.30	24,768.75	.002
	.07	1	1,768.328	42.127	.024	42,794,881.20	131,372.56	.003
	.07	2	1,075.273	8.607	.008	42,794,881.20	131,372.56	.003
	.07	4	1,015.318	10.543	.010	42,794,881.20	131,372.56	.003
	.10	1	10,550.762	254.201	.024	230,401,556.50	440,500.61	.002
	.10	2	6,379.845	49.041	.008	230,401,556.50	440,500.61	.002
	.10	4	6,068.725	51.074	.008	230,401,556.50	440,500.61	.002

APPENDIX C

DDMCE: Running times for Dynamic Graphs

			Time			Cliques		
n	ρ	c	$\bar{x}$	σ^2	$\sigma^2/\bar{x}$	$\bar{x}$	σ^2	$\sigma^2/\bar{x}$
1,000	.01	1	.029	.006	.206	18.10	12.07	.667
	.01	2	.021	.003	.127	18.10	12.07	.667
	.01	4	.022	.002	.109	18.10	12.07	.667
	.03	1	.061	.010	.162	141.30	43.58	.308
	.03	2	.043	.009	.215	141.30	43.58	.308
	.03	4	.043	.010	.228	141.30	43.58	.308
	.05	1	.119	.013	.111	547.80	110.09	.201
	.05	2	.098	.013	.132	547.80	110.09	.201
	.05	4	.101	.015	.152	547.80	110.09	.201
	.07	1	.210	.032	.150	1,479.00	373.68	.253
	.07	2	.164	.023	.139	1,479.00	373.68	.253
	.07	4	.160	.024	.150	1,479.00	373.68	.253
	.10	1	.483	.041	.084	4,614.20	873.73	.189
	.10	2	.388	.040	.102	4,614.20	873.73	.189
	.10	4	.365	.051	.141	4,614.20	873.73	.189
	.15	1	1.037	.074	.071	25,820.00	3,633.32	.141
	.15	2	.906	.062	.069	25,820.00	3,633.32	.141
	.15	4	1.015	.125	.123	25,820.00	3,633.32	.141
	.20	1	2.360	.223	.094	117,226.10	18,193.51	.155
	.20	2	1.852	.178	.096	117,226.10	18,193.51	.155
	.20	4	2.069	.145	.070	117,226.10	18,193.51	.155
	.25	1	8.501	.616	.072	539,246.20	46,820.70	.087
	.25	2	5.381	.421	.078	539,246.20	46,820.70	.087
	.25	4	5.665	.446	.079	539,246.20	46,820.70	.087

			Time			Cliques		
n	ρ	c	$\bar{x}$	σ^2	$\sigma^2/\bar{x}$	$\bar{x}$	σ^2	$\sigma^2/\bar{x}$
	.30	1	40.488	6.417	.158	2,350,002.50	263,813.37	.112
	.30	2	22.151	2.438	.110	2,350,002.50	263,813.37	.112
	.30	4	21.711	2.341	.108	2,350,002.50	263,813.37	.112
5,000	.01	1	.125	.015	.120	526.50	127.99	.243
	.01	2	.099	.007	.069	526.50	127.99	.243
	.01	4	.100	.008	.078	526.50	127.99	.243
	.03	1	.541	.054	.101	6,852.10	1,378.91	.201
	.03	2	.449	.063	.141	6,852.10	1,378.91	.201
	.03	4	.432	.062	.144	6,852.10	1,378.91	.201
	.05	1	1.267	.205	.162	45,015.00	6,977.27	.155
	.05	2	1.103	.160	.145	45,015.00	6,977.27	.155
	.05	4	1.102	.118	.107	45,015.00	6,977.27	.155
	.07	1	3.854	.360	.094	148,254.00	14,751.70	.100
	.07	2	2.804	.255	.091	148,254.00	14,751.70	.100
	.07	4	2.746	.162	.059	148,254.00	14,751.70	.100
	.10	1	19.324	1.704	.088	880,500.50	80,173.54	.091
	.10	2	11.875	1.055	.089	880,500.50	80,173.54	.091
	.10	4	11.580	.931	.080	880,500.50	80,173.54	.091
10,000	.01	1	.292	.049	.166	1,704.00	411.13	.241
	.01	2	.228	.033	.145	1,704.00	411.13	.241
	.01	4	.194	.035	.180	1,704.00	411.13	.241
	.03	1	1.788	.133	.074	53,421.20	4,502.01	.084
	.03	2	1.405	.131	.093	53,421.20	4,502.01	.084
	.03	4	1.593	.114	.071	53,421.20	4,502.01	.084
	.05	1	11.138	.917	.082	292,509.80	26,858.74	.092
	.05	2	7.247	.647	.089	292,509.80	26,858.74	.092
	.05	4	6.687	.533	.080	292,509.80	26,858.74	.092
	.07	1	59.519	4.347	.073	1,519,681.60	114,156.32	.075
	.07	2	37.237	2.667	.072	1,519,681.60	114,156.32	.075
	.07	4	34.676	2.517	.073	1,519,681.60	114,156.32	.075
	.10	1	491.325	26.862	.055	11,486,101.10	686,248.56	.060
	.10	2	301.977	17.171	.057	11,486,101.10	686,248.56	.060
	.10	4	282.626	16.634	.059	11,486,101.10	686,248.56	.060

Bibliography

[1] Pauchet A., Rioult F., Chanoni E., Ales Z., and Şerban O. Advances on dialogue modelling interactive narration requires prominent interaction and emotion. In Joaquim Filipe and Ana Fred, editors, *Proceedings of the 5th International Conference on Agents and Artificial Intelligence*, volume 1, pages 527–530. SciTePress, 2013.

[2] Steven Abney. Parsing by chunks. *Principle-based parsing*, 44:257–278, 1991.

[3] Amel Achour, Marc Le Tallec, Sébastien Saint-Aimé, Brigitte Le Pévédic, Jeanne Villaneau, J-Y Antoine, and Dominique Duhaut. Emotirob: from understanding to cognitive interaction. In *Mechatronics and Automation, 2008. ICMA 2008. IEEE International Conference on*, pages 369–374. IEEE, 2008.

[4] C.C. Aggarwal, J.B. Orlin, and R.P. Tai. Optimized crossover for the independent set problem. *Operations Research*, 45(2):226–234, 1997.

[5] EA Akkoyunlu. The enumeration of maximal cliques of large graphs. *SIAM Journal on Computing*, 2:1, 1973.

[6] Z. Ales, G. Dubuisson Duplessis, O. Şerban, and A. Pauchet. A methodology to design human-like embodied conversational agents. *Proceedings of the 1st International Workshop on Human-Agent Interaction Design and Models*, pages 34–49, 2012.

[7] V. Alfred. Algorithms for finding patterns in strings. *Handbook of Theoretical Computer Science: Algorithms and complexity*, 1:255, 1990.

[8] Jens Allwood. An activity based approach to pragmatics. *Abduction, belief and context in dialogue: Studies in computational pragmatics*, pages 47–80, 2000.

[9] C.O. Alm, D. Roth, and R. Sproat. Emotions from text: machine learning for text-based emotion prediction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 579–586. Association for Computational Linguistics, 2005.

[10] Apple. Siri. `http://www.apple.com/ios/siri/`, March 2013.

[11] S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Seventh conference on International Language Resources and Evaluation, Malta. Retrieved May*, volume 25, page 2010, 2010.

[12] E. Balas and W. Niehaus. Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems. *Journal of Heuristics*, 4(2):107–122, 1998.

[13] O. Bälter, O. Engwall, A.-M. Öster, and H. Kjellström. Wizard-of-oz test of artur: a computer-based speech training system with articulation correction. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, pages 36–43. ACM, 2005.

[14] A. Baylor and J. Ryu. The api (agent persona instrument) for assessing pedagogical agent persona. In *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, volume 2003, pages 448–451, 2003.

[15] R. Beale and C. Creed. Affective interaction: How emotional agents affect users. *International Journal of Human-Computer Studies*, 67(9):755–776, 2009.

[16] C. Benzmüller, H. Horacek, I. Kruijff-Korbayová, H. Lesourd, M. Schiller, and M. Wolska. Diawoz-ii–a tool for wizard-of-oz experiments in mathematics. In *KI 2006: Advances in Artificial Intelligence*, pages 159–173. Springer, 2007.

[17] A. Bersoult. Les enfants de 7 ans interagissent-ils avec un agent virtual comme evec un adulte ? un situation de narration. Master's thesis, Universite de Rouen, 2012.

[18] Timothy W Bickmore and Rosalind W Picard. Establishing and maintaining long-term human-computer relationships. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 12(2):293–327, 2005.

[19] S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G. A. Reis, and J. Reynar. Building a sentiment summarizer for local service reviews. In *WWW Workshop on NLP in the Information Explosion Era*, 2008.

[20] R.C. Bolles. Robust feature matching through maximal cliques. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 182, page 140, 1979.

[21] I.M. Bomze, M. Budinich, P.M. Pardalos, and M. Pelillo. The maximum clique problem. *Handbook of combinatorial optimization*, 4(1):1–74, 1999.

[22] I. Borg and P.J.F. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Verlag, 2005.

[23] U. Brandes and T. Erlebach. *Network analysis: Methodological foundations*, volume 3418. Springer Verlag, 2005.

[24] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[25] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern recognition letters*, 19(3):255–259, 1998.

[26] D. C. Burnett, M. R. Walker, and A. Hunt. Speech synthesis markup language (ssml) version 1.0. `http://www.w3.org/TR/speech-synthesis/`, September 2009.

[27] C. C. Russell and M. K. Odell. Soundex. `http://www.archives.gov/research/census/soundex.html`, March 2013.

[28] R.A. Calvo and S. D'Mello. Affect detection: An interdisciplinary review of models, methods, and their applications. *IEEE Transactions on Affective Computing*, pages 18–37, 2010.

[29] R. Carraghan and P.M. Pardalos. An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9(6):375–382, 1990.

[30] J. Cassell. Nudge nudge wink wink: Elements of face-to-face conversation for embodied conversational agents. *Embodied conversational agents*, pages 1–27, 2000.

[31] Marc Cavazza, Raul Santos de la Camara, and Markku Turunen. How was your day?: a companion eca. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1629–1630. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[32] F. Cazals and C. Karande. A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1-3):564–568, 2008.

[33] CERN. The colt distribution: Open source libraries for high performance scientific and technical computing in java, April 2012. CERN - European Organization for Nuclear Research.

[34] C. Chang, S. Chatterjee, and P.R. Kube. On an analysis of static occlusion in stereo vision. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 722–723. IEEE, 1991.

[35] F. R. Chaumartin. Upar7: A knowledge-based system for headline sentiment tagging. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 422–425, 2007.

[36] W.J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):749–764, 1995.

[37] Herbert H Clark. *Using language*, volume 4. Cambridge University Press Cambridge, 1996.

[38] CMLabs. Psyclone. `http://www.mindmakers.org/projects/psyclone/`, 2007.

[39] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[40] Matthieu Courgeon, Jean-Claude Martin, and Christian Jacquemin. Marc: a multimodal affective and reactive character. In *Proceedings of the 1st Workshop on AFFective Interaction in Natural Environments*, 2008.

[41] Roddy Cowie. Describing the forms of emotional colouring that pervade everyday life. *Oxford handbook of philosophy of emotion. Oxford University Press, London*, pages 63–94, 2010.

[42] Roddy Cowie, Gary McKeown, and Ellen Douglas-Cowie. Tracing emotion: an overview. *International Journal of Synthetic Emotions (IJSE)*, 3(1):1–17, 2012.

[43] R. Cox. Regular expression matching can be simple and fast (but is slow in java, perl, php, python, ruby, ...). `http://swtch.com/~rsc/regexp/regexp1.html`, January 2007.

[44] T. Danisman and A. Alpkocak. Feeler: Emotion classification of text using vector space model. In *AISB 2008 Convention Communication, Interaction and Social Intelligence*, volume 1, page 53, 2008.

[45] C. Darwin, P. Ekman, and P. Prodger. *The expression of the emotions in man and animals*. Oxford University Press, USA, 2002.

[46] Nikolaos Dimakis, John K Soldatos, Lazaros Polymenakos, Pascal Fleury, Jan Curín, and Jan Kleindienst. Integrated development of context-aware applications in smart spaces. *Pervasive Computing, IEEE*, 7(4):71–79, 2008.

[47] S.K. D'Mello, S.D. Craig, J. Sullins, and A.C. Graesser. Predicting affective states expressed through an emote-aloud procedure from AutoTutor's mixed-initiative dialogue. *International Journal of Artificial Intelligence in Education*, 16(1):3–28, 2006.

[48] E. Douglas-Cowie, R. Cowie, C. Cox, N. Amier, and D. Heylen. The sensitive artificial listner: an induction technique for generating emotionally coloured conversation. *Preceedings of Workshop on Corpora for Research on Emotion and Affect*, 2008.

[49] E. Douglas-Cowie, R. Cowie, I. Sneddon, C. Cox, O. Lowry, M. Mcrorie, J.C. Martin, L. Devillers, S. Abrilian, A. Batliner, et al. The HUMAINE database: addressing the collection and annotation of naturalistic and induced emotional data. *Affective computing and intelligent interaction*, pages 488–500, 2007.

[50] N. Du, B. Wu, L. Xu, B. Wang, and X. Pei. A parallel algorithm for enumerating all maximal cliques in complex network. In *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*, pages 320–324. IEEE, 2006.

[51] G. Dubuisson Duplessis, N. Chaignaud, J-Ph. Kotowicz, A. Pauchet, and J-P. Pécuchet. Empirical specification of dialogue games for an interactive agent. In Y. Demazeau, T. Ishida, J.M. Corchado, and J. Bajo, editors, *Proceedings of the 11th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*, volume LNAI 7879, pages 49–60, May 2013.

[52] Ken Ducatel, Marc Bogdanowicz, Fabiana Scapolo, Jos Leijten, and Jean-Claude Burgelman. *Scenarios for ambient intelligence in 2010*. Office for official publications of the European Communities, 2001.

[53] S.T. Dumais. Latent semantic analysis. *Annual Review of Information Science and Technology*, 38(1):188–230, 2004.

[54] P. Ekman et al. *Universals and cultural differences in facial expressions of emotion*. University of Nebraska Press, 1971.

[55] P. Ekman et al. An argument for basic emotions. *Cognition and emotion*, 6(3/4):169–200, 1992.

[56] P. Ekman and W.V. Friesen. The repertoire of nonverbal behavior: Categories, origins, usage, and coding. *Semiotica*, 1(1):49–98, 1969.

[57] P. Ekman and W.V. Friesen. Facial Action Coding System (FACS): A technique for the measurement of facial action. *Palo Alto, CA: Consulting*, 1978.

[58] P. Ekman and E.L. Rosenberg. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, USA, 2005.

[59] Paul Ekman. Basic emotions. *Handbook of cognition and emotion*, 98:45–60, 1999.

[60] F. Eyben, M. Wöllmer, and B. Schuller. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the international conference on Multimedia*, pages 1459–1462. ACM, 2010.

[61] C. Faloutsos and D.W. Oard. A survey of information retrieval and filtering methods. *University of Maryland Computer Science Department*, 1:23, 1998.

[62] C. Fellbaum et al. Wordnet and wordnets. *Encyclopedia of Language and Linguistics*, pages 665–670, 2005.

[63] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. *PVM: Parallel virtual machine: a users\'guide and tutorial for networked parallel computing*. MIT Press, 1994.

[64] Rodolphe Gelin, Christophe d'Alessandro, Q. Anh Le, Olivier Deroo, David Doukhan, Jean-Claude Martin, Catherine Pelachaud, Albert Rilliard, and Sophie Rosset. Towards a storytelling humanoid robot. In *AAAI Fall Symposium Series*, 2010.

[65] A.M. Glenberg, D. Havas, R. Becker, and M. Rinck. Grounding language in bodily states. *Grounding cognition: the role of perception and action in memory, language, and thinking*, page 115, 2005.

[66] N. Godbole, M. Srinivasaiah, and S. Skiena. Large-scale sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, volume 2, 2007.

[67] Google. Protocol buffers. `https://developers.google.com/protocol-buffers/`, 2012.

[68] Google. Speech recognition api. `http://developer.android.com/reference/android/speech/package-summary.html`, March 2013.

[69] D. Greene, M. Parnas, and F. Yao. Multi-index hashing for information retrieval. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 722–731. IEEE, 1994.

[70] A. Grosso, M. Locatelli, and F.D. Croce. Combining swaps and node weights in an adaptive greedy approach for the maximum clique problem. *Journal of Heuristics*, 10(2):135–152, 2004.

[71] A. Grosso, M. Locatelli, and W. Pullan. Simple ingredients leading to very efficient heuristics for the maximum clique problem. *Journal of Heuristics*, 14(6):587–612, 2008.

[72] O. Grynszpan, J.-C. Martin, and J. Nadel. Multimedia interfaces for users with high functioning autism: An empirical investigation. *International Journal of Human-Computer Studies*, 66(8):628–639, 2008.

[73] H. Gunes and B. Schuller. Categorical and dimensional affect analysis in continuous input: Current trends and future directions. *Image and Vision Computing*, 2012.

[74] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[75] J. Han, M. Jo, S. Park, and S. Kim. The educational use of home robots for children. In *Robot and Human Interactive Communication*, pages 378–383. IEEE, 2005.

[76] E.R. Hancock and J. Kittler. Edge-labeling using dictionary-based relaxation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(2):165–181, 1990.

[77] Gerd Herzog, Alassane Ndiaye, Stefan Merten, Heinz Kirchmann, Tilman Becker, and Peter Poller. Large-scale software integration for spoken language and multimodal dialog systems. *Natural Language Engineering*, 10(3-4):283–305, 2004.

[78] D. Heylen, S. Kopp, S. Marsella, C. Pelachaud, and H. Vilhjalmsson. Why conversational agents do what they do? functional representations for generating conversational agent behavior. In *Proceedings of the First Functional Markup Language Workshop*, 2008.

[79] P. Hintjens. *Zeromq: Messaging for Many Applications*. O'Reilly Media, 2013.

[80] R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximal cliques. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(11):1168–1180, 1989.

[81] Hung-Hsuan Huang, Aleksandra Cerekovic, Kateryna Tarasenko, Vjekoslav Levacic, Goranka Zoric, Igor S Pandzic, Yukiko Nakano, and Toyoaki Nishida. Integrating embodied conversational agent components with a generic framework. *Multiagent and Grid Systems*, 4(4):371–386, 2008.

[82] Hung-Hsuan Huang, Aleksandra Cerekovic, Kateryna Tarasenko, Vjekoslav Levacic, Goranka Zoric, Margus Treumuth, Igor S Pandzic, Yukiko Nakano, and Toyoaki Nishida. An agent based multicultural user interface in a customer service application. In *The eNTERFACEâĂŹ06 international workshop on multimodal interfaces. Dubrovnik, Croatia*, 2006.

[83] Hung-Hsuan Huang, Kateryna Tarasenko, Toyoaki Nishida, Aleksandra Cerekovic, Vjekoslav Levacic, Goranka Zoric, Igor S Pandzic, and Yukiko Nakano. An agent based multicultural tour guide system with nonverbal user interface. *Journal on Multimodal User Interfaces*, 1(1):41–48, 2007.

[84] David Huggins-Daines, Mohit Kumar, Arthur Chan, Alan W Black, Mosur Ravishankar, and Alex I Rudnicky. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I. IEEE, 2006.

[85] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 241–250. ACM, 2008.

[86] HUMAINE Community Portal. http://emotion-research.net/, March 2011.

[87] A. Hunt and S. McGlashan. Speech recognition grammar specification version 1.0. http://www.w3.org/TR/speech-grammar/, March 2004.

[88] Ji Hyungsuk, S. Ploux, and E. Wehrli. Lexical knowledge representation with contextonyms. In *Proceedings of the 2003 MT Summit IX*. Association for Machine Translation in the Americas, 2003.

[89] Pavel Ircing, Jan Romportl, and Zdenek Loose. Audiovisual interface for czech spoken dialogue system. In *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pages 526–529. IEEE, 2010.

[90] iSpeech Inc. ispeech tts service. http://www.ispeech.org/, March 2013.

[91] K. Jaber and R.A. Nur'Aini Abdul Rashid. The parallel maximal cliques algorithm for protein sequence clustering. *American Journal of Applied Sciences*, 6(7):1368–1372, 2009.

[92] W. James. What is an Emotion? *Mind*, 9(34):188–205, 1884.

[93] E. Jennings and L. Motyčková. A distributed algorithm for finding all maximal cliques in a network graph. *LATIN'92: 1st Latin American Symposium on Theoretical Informatics, São Paulo, Brazil, April 6-10, 1992: proceedings*, 583:281–293, 1992.

[94] H. Ji, S. Ploux, and E. Wehrli. Lexical knowledge representation with contextonyms. In *Proceedings of MT Summit IX, New Orleans, USA*. Association for Machine Translation in the Americas, 2003.

[95] Y. Ji, X. Xu, and G.D. Stormo. A graph theoretical approach for predicting common rna secondary structure motifs including pseudoknots in unaligned sequences. *Bioinformatics*, 20(10):1591–1602, 2004.

[96] D.S. Johnson and M.A. Trick. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993*, volume 26. Amer Mathematical Society, 1996.

[97] M. Johnston, P. Baggia, D. C. Burnett, J. Carter, D. A. Dahl, G. McCobb, and D. Raggett. Emma: Extensible multimodal annotation markup language. http://www.w3.org/TR/emma/, February 2009.

[98] T. Kanda, T. Hirano, D. Eaton, and H. Ishiguro. Interactive robots as social partners and peer tutors for children: A field trial. *Human-Computer Interaction*, 19(1):61–84, 2004.

[99] R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 40(4):85–103, 1972.

[100] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13(3):637–649, 2001.

[101] J. Klein, Y. Moon, and R. W. Picard. This computer responds to user frustration:: Theory, design, and results. *Interacting with computers*, 14(2):119–140, 2002.

[102] S. R. Klemmer, A. K. Sinha, J. Chen, J. A Landay, N. Aboobaker, and A. Wang. Suede: a wizard of oz prototyping tool for speech user interfaces. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 1–10. ACM, 2000.

[103] I. Koch. Enumerating all connected maximal common subgraphs in two graphs. *Theoretical Computer Science*, 250(1-2):1–30, 2001.

[104] I. Koch, T. Lengauer, and E. Wanke. An algorithm for finding maximal common subtopologies in a set of protein structures. *Journal of computational biology*, 3(2):289–306, 1996.

[105] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[106] S. Kopp, B. Krenn, S. Marsella, A. Marshall, C. Pelachaud, H. Pirker, K. Thórisson, and H. Vilhjálmsson. Towards a common framework for multimodal generation: The behavior markup language. In *Intelligent Virtual Agents*, pages 205–217. Springer, 2006.

[107] Z. Kozareva, B. Navarro, S. Vázquez, and A. Montoyo. Ua-zbsa: A headline emotion classification through web information. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 334–337. Association for Computational Linguistics, 2007.

[108] H. Kozima, C. Nakagawa, and Y. Yasuda. Interactive robots for communication-care: a case-study in autism therapy. In *Robot and Human Interactive Communication*, pages 341–346, 2005.

[109] Paul Lamere, Philip Kwok, William Walker, Evandro Gouvea, Rita Singh, Bhiksha Raj, and Peter Wolf. Design of the cmu sphinx-4 decoder. In *Proceedings of the 8th European conference on speech communication and technology*, pages 1181–1184, 2003.

[110] Jean-Yves Lionel Lawson, Ahmad-Amr Al-Akkad, Jean Vanderdonckt, and Benoit Macq. An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 245–254. ACM, 2009.

[111] Akinobu Lee, Tatsuya Kawahara, and Kiyohiro Shikano. Julius: an open source real-time large vocabulary recognition engine. `http://julius.sourceforge.jp/`, August 2012.

[112] B. Liu. Sentiment analysis and subjectivity. *Handbook of Natural Language Processing,*, pages 627–666, 2010.

[113] H. Liu and P. Singh. ConceptNet-a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004.

[114] Hugo Liu, Henry Lieberman, and Ted Selker. A model of textual affect sensing using real-world knowledge. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 125–132. ACM, 2003.

[115] R Duncan Luce and Albert D Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

[116] C. Ma, H. Prendinger, and M. Ishizuka. A chat system based on emotion estimation from text and embodied conversational messengers. *Entertainment Computing-ICEC 2005*, pages 535–538, 2005.

[117] K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. *Algorithm Theory*, 9:260–272, 2004.

[118] G. Malewicz, M.H. Austern, A.J.C. Bik, J.C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 international conference on Management of data*, pages 135–146. ACM, 2010.

[119] G.S. Manku, A. Jain, and A. Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web*, pages 141–150. ACM, 2007.

[120] T. Meltzer, C. Yanover, and Y. Weiss. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 428–435. IEEE, 2005.

[121] R. Mihalcea and D. Radev. *Graph-based natural language processing and information retrieval*. Cambridge University Press, 2011.

[122] J. Miksatko, K. H. Kipp, and M. Kipp. The persona zero-effect: Evaluating virtual character benefits on a learning task with repeated interactions. In *Intelligent Virtual Agents*, pages 475–481. Springer, 2010.

[123] G.A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[124] A. Mohaisen, A. Yun, and Y. Kim. Measuring the mixing time of social graphs. In *Proceedings of the 10th annual conference on Internet measurement*, pages 383–389. ACM, 2010.

[125] R. Mohr and T.C. Henderson. Arc and path consistency revisited. *Artificial intelligence*, 28(2):225–233, 1986.

[126] L. P. Morency, R. Mihalcea, and P. Doshi. Towards multimodal sentiment analysis: harvesting opinions from the web. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 169–176. ACM, 2011.

[127] M. Mori. The uncanny valley. *Energy*, 7(4):33–35, 1970.

[128] M. Moundridou and M. Virvou. Evaluating the persona effect of an interface agent in a tutoring system. *Journal of computer assisted learning*, 18(3):253–261, 2002.

[129] C. Munteanu and M. Boldea. Mdwoz: A wizard of oz environment for dialog systems development. In *LRECâĂŹ00*. Citeseer, 2000.

[130] N.M. Nasrabadi and C.Y. Choo. Hopfield network for stereo vision correspondence. *Neural Networks, IEEE Transactions on*, 3(1):5–13, 1992.

[131] Alena Neviarouskaya, Helmut Prendinger, and Mitsuru Ishizuka. Textual affect sensing for sociable and expressive online communication. In *Affective Computing and Intelligent Interaction*, pages 218–229. Springer, 2007.

[132] NIST II Smart space library. `http://www.nist.gov/smartspace/`, March 2013.

[133] Nuance. Dragon speech recognition software. `http://www.nuance.com/dragon/index.htm`, March 2013.

[134] M. Ochs, J. Ollivier, B. Coic, T. Brien, and F. Majeric. Affimo: Toward an open-source system to detect affinities and emotions in user's sentences. In *Workshop Affect, Compagnon Artificiel, Interaction (WACAI), Grenoble, France*, volume 1, 2012.

[135] M. Ochs and H. Prendinger. A virtual characterâĂŹs emotional persuasiveness. In *Kansei Engineering and Emotion Research International Conference*, 2010.

[136] Jeff Orkin and Deb Roy. Automatic learning and generation of social behavior from collective human gameplay. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 385–392. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[137] C.E. Osgood, W.H. May, and M.S. Miron. *Cross-cultural universals of affective meaning*. University of Illinois Press, 1975.

[138] C.E. Osgood, G.J. Suci, and P.H. Tannenbaum. *The measurement of meaning*. University of Illinois Press, 1971.

[139] Alexander Osherenko and Elisabeth André. Differentiated semantic analysis in lexical affect sensing. In *Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on*, pages 1–6. IEEE, 2009.

[140] M. Otto, R. Friesen, and D. Rösner. Message oriented middleware for flexible wizard of oz experiments in hci. In *Human-Computer Interaction. Design and Development Approaches*, pages 121–130. Springer, 2011.

[141] S. Oviatt. Talking to thimble jellies: ChildrenâĂŹs conversational speech with animated characters. In *Proc. ICSLP (Beijing, China)*, pages 67–70, 2000.

[142] Oxford Dictionary, Online Edition. http://oxforddictionaries.com/, March 2011.

[143] P. P. Leach, M. Mealling, and R. Salz. A universally unique identifier (uuid) urn namespace. `http://www.ietf.org/rfc/rfc4122.txt`, July 2005.

[144] Sathish Pammi, Marcela Charfuelan, and Marc Schröder. Multilingual voice creation toolkit for the mary tts platform. *Proc. LREC. Valettea, Malta: ELRA*, 2010.

[145] P.M. Pardalos, J. Rappe, and M.G.C. Resende. An exact parallel algorithm for the maximum clique problem. *High performance algorithms and software in nonlinear optimization*, 24:279, 1998.

[146] P.A. Pevzner, S.H. Sze, et al. Combinatorial approaches to finding subtle signals in dna sequences. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, volume 8, pages 269–278, 2000.

[147] L. Philips. Metaphone 3. `http://www.amorphics.com/`, March 2013.

[148] Lawrence Philips. Hanging on the metaphone. *Computer Language*, 7(12 (December)), 1990.

[149] Lawrence Philips. The double metaphone search algorithm. *CC Plus Plus Users Journal*, 18(6):38–43, 2000.

[150] Rosalind W Picard and Jonathan Klein. Computers that recognise and respond to user emotion: theoretical and practical implications. *Interacting with computers*, 14(2):141–169, 2002.

[151] R.W. Picard. *Affective computing*. The MIT press, 2000.

[152] Yoann Pigné, Antoine Dutot, Frédéric Guinand, and Damien Olivier. Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. *arXiv preprint arXiv:0803.2093*, 2008.

[153] J. C. Platt et al. Using analytic qp and sparseness to speed training of support vector machines. *Advances in neural information processing systems*, pages 557–563, 1999.

[154] I. Poggi, C. Pelachaud, F. Rosis, V. Carofiglio, and B. Carolis. Greta. a believable embodied conversational agent. *Multimodal intelligent information presentation*, pages 3–25, 2005.

[155] Livia Polanyi and Annie Zaenen. Contextual valence shifters. In *Computing attitude and affect in text: Theory and applications*, pages 1–10. Springer, 2006.

[156] Christopher Potts. Sentiment analysis tutorial, November 2011.

[157] H. Prendinger and M. Ishizuka. The empathic companion: A character-based interface that addresses users'affective states. *Applied Artificial Intelligence*, 19(3-4):267–285, 2005.

[158] H. Prendinger, S. Mayer, J. Mori, and M. Ishizuka. Persona effect revisited. In *Intelligent Virtual Agents*, pages 283–291. Springer, 2003.

[159] K. Prepin and C. Pelachaud. Basics of intersubjectivity dynamics: Model of synchrony emergence when dialogue partners understand each other. In *Agents and Artificial Intelligence*, pages 302–318. Springer, 2013.

[160] W. Pullan and H.H. Hoos. Dynamic local search for the maximum clique problem. *Journal of Artificial Intelligence Research*, 25(1):159–185, 2006.

[161] D. Rao and D. Ravichandran. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 675–682. Association for Computational Linguistics, 2009.

[162] V. Rieser and O. Lemon. Learning effective multimodal dialogue strategies from wizard-of-oz data: Bootstrapping and evaluation. In *Proceedings of ACL*, pages 638–646, 2008.

[163] B. Robins, P. Dickerson, and K. Dautenhahn. Robots as embodied beings - interactionally sensitive body movements in interactions among autistic children and a robot. In *Robot and Human Interactive Communication*, pages 54–59, 2005.

[164] A. Rosenfeld, R.A. Hummel, and S.W. Zucker. Scene labeling by relaxation operations. *Systems, Man and Cybernetics, IEEE Transactions on*, 6(6):420–433, 1976.

[165] J.A. Russell. Core affect and the psychological construction of emotion. *Psychological review*, 110(1):145–172, 2003.

[166] K. Ryokai, C. Vaucelle, and J. Cassell. Virtual peers as partners in storytelling and literacy learning. *Journal of computer assisted learning*, 19(2):195–208, 2003.

[167] Sébastien Saint-Aimé, Brigitte Le-Pevedic, Dominique Duhaut, and Takanori Shibata. Emotirob: companion robot project. In *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, pages 919–924. IEEE, 2007.

[168] D. Salber and J. Coutaz. Applying the wizard of oz technique to the study of multimodal systems. *Human-Computer Interaction*, pages 219–230, 1993.

[169] R. Samudrala and J. Moult. A graph-theoretic algorithm for comparative modeling of protein structure1. *Journal of molecular biology*, 279(1):287–302, 1998.

[170] B. Santorini. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). Technical report, University of Pennsylvania, 1990.

[171] Klaus R Scherer. Emotions are emergent processes: they require a dynamic computational architecture. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1535):3459–3474, 2009.

[172] Klaus R Scherer, Elizabeth Clark-Polner, and Marcello Mortillaro. In the eye of the beholder? universality and cultural specificity in the expression and perception of emotion. *International Journal of Psychology*, 46(6):401–435, 2011.

[173] Albrecht Schmidt. Interactive context-aware systems interacting with ambient intelligence. *Ambient intelligence*, pages 159–178, 2005.

[174] M.C. Schmidt, N.F. Samatova, K. Thomas, and B.H. Park. A scalable, parallel algorithm for maximal clique enumeration. *Journal of Parallel and Distributed Computing*, 69(4):417–428, 2009.

[175] M. Schröder. The semaine api: towards a standards-based framework for building emotion-oriented systems. *Advances in Human-Computer Interaction*, 2010:2–2, 2010.

[176] M. Schröder. *The SEMAINE API: A component integration framework for a naturally interacting and emotionally competent Embodied Conversational Agent*. PhD thesis, Saarland University, 2011.

[177] M. Schröder, P. Baggia, F. Burkhardt, J.-C. Martin, C. Pelachaud, C. Peter, B. Schuller, I. Wilson, and E. Zovato. Elements of an emotionml 1.0. `http://www.w3.org/2005/Incubator/emotion/XGR-emotionml-20081120/`, November 2008.

[178] Marc Schröder, Sathish Pammi, and Oytun Türk. Multilingual mary tts participation in the blizzard challenge 2009. In *Proc. Blizzard Challenge*, volume 9, 2009.

[179] Björn Schuller, Michel Valster, Florian Eyben, Roddy Cowie, and Maja Pantic. Avec 2012: the continuous audio/visual emotion challenge. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pages 449–456. ACM, 2012.

[180] P. Singh, T. Lin, E. Mueller, G. Lim, T. Perkins, and W. Li Zhu. Open mind common sense: Knowledge acquisition from the general public. *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, pages 1223–1237, 2002.

[181] M. Snir, S.W. Otto, D.W. Walker, J. Dongarra, and S. Huss-Lederman. *MPI: The complete reference*. MIT press, 1995.

[182] B. Snyder, D. Bosanac, and R. Davies. *ActiveMQ in Action*. Manning Publications, 2011.

[183] Mohammad Soleymani, Maja Pantic, and Thierry Pun. Multimodal emotion recognition in response to videos. *Affective Computing, IEEE Transactions on*, 3(2):211–223, 2012.

[184] C. Solnon and S. Fenet. A study of aco capabilities for solving the maximum clique problem. *Journal of Heuristics*, 12(3):155–180, 2006.

[185] V. Stix. Finding all maximal cliques in dynamic graphs. *Computational Optimization and applications*, 27(2):173–186, 2004.

[186] Philip J Stone, Dexter C Dunphy, and Marshall S Smith. *The general inquirer: a computer approach to content analysis*. The MIT Press, 1966.

[187] C. Strapparava and R. Mihalcea. Learning to identify emotions in text. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1556–1560. ACM, 2008.

[188] C. Strapparava and A. Valitutti. WordNet-Affect: an affective extension of WordNet. In *Proceedings of LREC*, volume 4, pages 1083–1086. Citeseer, 2004.

[189] Dag Sverre Syrdal, Michael L Walters, Nuno Otero, Kheng Lee Koay, and Kerstin Dautenhahn. He knows when you are sleeping-privacy and the personal robot companion. In *Proc. Workshop Human Implications of Human-Robot Interaction, Association for the Advancement of Artificial Intelligence (AAAIâĂŹ07)*, pages 28–33, 2007.

[190] Y. R. Tausczik and J. W. Pennebaker. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of Language and Social Psychology*, 29(1):24–54, 2010.

[191] W. Taymans, S. Baker, A. Wingo, R. Bultje, and S. Kost. Gstreamer application development manual, 2001.

[192] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558, 2010.

[193] Kristinn R Thórisson, Hrvoje Benko, Denis Abramov, Andrew Arnold, Sameer Maskey, and Aruchunan Vaseekaran. Constructionist design methodology for interactive intelligences. *AI Magazine*, 25(4):77, 2004.

[194] Erik F Tjong Kim Sang and Sabine Buchholz. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics, 2000.

[195] Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

[196] E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42, 2006.

[197] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6:505, 1977.

[198] A. Valitutti, C. Strapparava, and O. Stock. Lexical resources and semantic similarity for affective evaluative expressions generation. *Affective Computing and Intelligent Interaction*, pages 474–481, 2005.

[199] VHMsg Library. `https://confluence.ict.usc.edu/display/VHTK/VHMsg`, March 2013.

[200] C. Von Hofsten and K. Rosander. *From action to cognition*, volume 164. Elsevier Science, 2007.

[201] VoxForge. The open source acoustic model. http://www.voxforge.org/.

[202] Wolfgang Wahlster. *Verbmobil: foundations of speech-to-speech translation*. Springer verlag, 2000.

[203] Wolfgang Wahlster. *SmartKom: Foundations of Multimodal Dialogue Systems (Cognitive Technologies)*. Springer-Verlag New York, Inc., 2006.

[204] H.G. Wallbott, K.R. Scherer, et al. Emotion and economic development-Data and speculations concerning the relationship between economic factors and emotional experience. *European journal of social psychology*, 18(3):267–273, 1988.

[205] S. Whittaker, M. Walker, and J. Moore. Fish or fowl: A wizard of oz evaluation of dialogue strategies in the restaurant domain. In *Language Resources and Evaluation Conference*, 2002.

[206] Michael Widenius, David Axmark, and AB MySQL. *MySQL reference manual: documentation from the source*. O'Reilly Media, Incorporated, 2002.

[207] J. Wiebe, T. Wilson, and C. Cardie. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2):165–210, 2005.

[208] Steve Young, Gunnar Evermann, Dan Kershaw, Gareth Moore, Julian Odell, Dave Ollason, Valtcho Valtchev, and Phil Woodland. *The HTK book*, volume 3. Cambridge University Engineering Department, 2002.

[209] Li Zhang. Exploration of affect sensing from speech and metaphorical text. In *Learning by Playing. Game-Based Education System Design and Development*, pages 251–262. Springer, 2009.