Relazione del progetto di Sistemi Multi-Agente LM

Applicazione delle tecniche di Text Mining all'interno di algoritmi di clustering per articoli scientifici.

Progetto realizzato da Michele Pratiffi

Con la supervisione del Professor Andrea Omicini e di Stefano Mariani

Capitolo 1

Introduzione al progetto

La domanda a cui il progetto intende rispondere è come possono essere sfruttate le tecniche di Text Mining all'interno di sistemi che hanno come obiettivo principale l'auto-organizzazione della conoscenza? Il progetto prevede la implementazione di un sistema di clustering per articoli scientifici in lingua inglese, sfruttando tecniche e algoritmi di Text Mining. Come infrastruttura di coordinazione si è scelto di utilizzare TuCSoN (Tuple Centres Spread over the Network) che è un modello per la cordinazione di processi distribuiti (agenti) che sfrutta i centri di tuple. All'interno del progetto sono stati realizzati due algoritmi di clustering uno ideato da me ideato e un'altro che implemanta un algoritmo preso dalla letteratura. Del secondo sono state implementate due varianti una prevede un calcolo poco distribuito (solo pochi agenti) mentre l'altra prevede una fortissima distribuzione del calcolo (un agente per articolo). Utilizzando questi algoritmi sono stati realizzati quindi molti e vari esperimenti, sia per la taratura dei parametri, che per tracciare la dipendenza degli algoritmi da questi. I risultati ottenuti verranno poi confrontati e analizzati nel dettaglio.

Capitolo 2

Analisi del Progetto

Come prima fase di questo step è stato necessario reperire e studiare diversi articoli riguardanti il Text Mining. In maniera tale di comprendere meglio cosa si intende per Text Mining, quali sono le tecniche oggi più frequentemente usate, quali siano le migliori e in che contesto è consigliato usare ogni singolo algoritmo. Poi è stato necessario analizzare i vari formati di rappresentazione degli articoli cosi da poter intuire quale tipologia è meglio utilizzare all'interno di un dato contesto. Esistono un'infinità di metodi per descrivere un articolo scientifico, nel progetto si è deciso di analizzare solo tre modelli il formato BibTex, l'abstract e l'articolo nella sua interezza. Si è pensato quindi inizialmente di realizzare vari esperimenti, utilizzando un semplice algoritmo di clustering, per scegliere quale di queste rappresentazioni fosse la migliore per il nostro problema. Una volta presa questa decisione verrano messi a punto gli altri algoritmi di clustering più complessi.

Capitolo 3

Realizzazione del progetto

3.0.1 Metodi di estrazione informazioni

Prima di tutto è necessario estrarre le informazioni dai vari formati di rappresentazione degli articoli. Tutti gli articoli utilizzati, in tutti i loro formati sono stati raccolti dal sito APICe che contiene una folta raccolta di testi di ricerca redatti o utilizzati dai ricercatori e professori della seconda facolta di ingegneria di Cesena. Il formato BibTex e gli abstract degli articoli sono stati scaricati e salvati come semplici file di testo txt, gli articoli in formato full text sono stati scaricati come pdf. Per ogni tiplogia di rappresentazione è stato implementato una semplice classe in Java che estrae dai testi le informazioni più interessati. Ognuna di queste classi legge le informazioni grezze le spezza nelle parti fondamentali e le organizza all'interno di apposite strutture dati. La struttura dati scelta è una mappa dove le chiavi sono i titoli degli articoli e i valori sono i relativi contenuti del testo.

Bibtex

Il file di testo contenente il BibTex, dato che su ogni riga si trova una coppia di valori tag valore, è stato letto riga per riga. Ogni riga viene trascritta in una stringa che successivamente subisce un processo di elaborazione per l'estrazione dei dati rilevanti. La stringa contenente il contenuto della riga viene quindi suddivisa in due stringhe una contenente il tag e l'altra il suo valore. Si controlla ora se il tag letto è di interesse o meno, se fa parte dell'insieme dei tag ritenuti poco rilevanti si passa a leggere e analizzire la coppia di valori successiva. Nel caso invece sia di interesse viene elaborata la stringa contenente il valore. Innanzi tutto vengono rimossi tutti i simbolismi come parentesi, punteggiatura varia a parte la virgola. Questo perché la virgola all'interno del formato BibTex viene utilizzata come separatore fra i vari valori del tag se ne esistono più di uno. Dato che spesso alcuni tag hanno come valori frasi, ad esempio il titolo dell'elaborato o le

note, viene applicato un filtro che elimina le stop word cioè le parole che non contengono concetti. Sono ritenute ad esempio stop word gli articoli, le preposizioni e le congiunzioni.

Abstracts

Gli abstract dei vari articoli sono sotto forma di testo non strutturato, scritto in inglese. Per ottenere le informazioni più significative, che altro non sono che le parole chiave, è sufficiente leggerne il contenuto e attuare tecniche varie di Text Mining come lo stemming e filtrare le stop word. Dal testo letto vengono rimossi tutti i simboli di punteggiatura, parentesi, caratteri speciali e numerici si ottiene così un testo composto solamente da parole. Si controlla poi se le parole sono stop word, in questo caso vengono direttamente rimosse. Viene quindi applicato un l'algoritmo di stemming che ha come obiettivo quello di accorpare le parole che hanno la stessa radice (stem) in uno stesso gruppo. Se un termine appare più di una volta si tiene traccia del numero di volte che appare che verra associato alla parola e inserito nella struttura dati.

Plain text

Dato che l'articolo nella sua interezza è in formato pdf è stato in primis necessario passare dal pdf a un semplice file di testo. Per questo passaggio è stata utilizzata una libreria java pdfbox che con un semplice comando trascrive in formato txt il pdf. L'argomento che viene passato al main della classe org.apache.pdfbox.ExtractText è il seguente:

-force <path assoluta file sorgente> <path assoluta file destinazione>

Una volta ottenuto il formato txt si procede alla lettura del contenuto dell'articolo come stringa di testo. Vengono poi rimossi punteggiatura, caratteri speciali, numeri. Si ottiene così un grandissimo insieme di parole, che devono essere filtrate in maniera tale da ridurne il numero. Innanzi tutto si è scelto di ignorare le parole composte da tre o meno caratteri dato che generalmente non sono rilevanti. Vengono poi eliminate dall'elenco tutte le stop words. È stato poi implementato e applicato l'algoritmo di stemming di Porter dopo un'attenta analisi si è deciso di non usare gli ultimi due passaggi dell'algoritmo dato che il numero di parole rimosse era basso a fronte di uno stravolgimento della loro forma. Per ogni parola si è tenuto conto delle sue occorrenze sulla base delle quali la lista di parole finali è statà ordinata dalla più alla meno frequente. Dato che l'elenco delle parole così trovate è molto grande, con dimensioni comprese fra le cento e le mille parole si è deciso dare la possibilità di selezionare solo le n parole più frequenti. Dove n è un parametro che si può settare a piacimento.

3.1 Esperimenti con BibTex, abstract o plain text

Esistono oggi molte raccolte di articoli di ricerca che mettono a disposizione vari formati, l'abstract dell'articolo, il testo in toto, la citazione in vari formati. Come primo esperimento ritengo sia quindi interessante provare ad organizzare gli articoli su diverse basi, in base al bibtex, in base all'abstract e infine in base al plain text. Verranno poi confrontati i risultati ottenuti in modo tale da scegliere quale delle rappresentazioni da i risultati migliori in termini di organizzazione e tempo di computazione. Come base dati per il test verranno utilizzati gli articoli presenti sul portale APICe che contiene un folto numero di articoli relativi alla ricerca svolta nella seconda facoltà di ingegneria di Cesena. Sono stati scelti circa cento articoli del portale APICe, che coprono l'ultimo periodo di ricerca dal 2010 al 2013. Per quanto riguarda il formato BibTex, innanzi tutto è necessario rimuovere i tag che non sono rilevanti come Pages, Isbn, Isbn-10, Chapter, la label, Doi, Issn, Volume, Url, Url-pdf, Pdf-local, Issn-online, Isbn-online, eccetera. Dato che in nel formato BibTex di alcuni articoli è presente anche l'abstract questo viene rimosso dato che lo si vuole analizzare successivamente da solo. Successivamente, in base al risultato che si vuole ottenere dalla organizzazione, vengono assegnati alle parole chiave i pesi, la somma dei pesi per un singolo articolo deve essere pari a 1. Ad ogni parola chiave dell'articolo viene dato un peso pari a 1/n dove n è il numero di parole chiave dell'articolo. Questa scelta è stata presa in maniera autonoma e non tratta dalla letteratura, pareva infatti essere questa la scelta migliore volendo in questo esperimento avere una funzione per l'assegnamento dei pesi troppo complessa. La funzione di correlazione semplicemente prende le tuple di due articoli e ne confronta una ad una le parole chiave, se una parola chiave esiste in entrambi gli articoli il suo peso viene moltiplicato per uno, nel caso contrario per zero. Il fattore di correlazione fra i due testi totale è pari alla somma dei pesi così ottenuti. Il fattore di correlazione ottenuto sarà così sempre compreso tra zero e uno. Il confronto fra le parole chiave viene attuato tramite un semplice match sintattico, che è quello fornito dalla classe alice.logictuple.TupleArgument.

3.1.1 Struttura dei centri di tuple e loro inizializzazione

Per l'esperimento in questione si è deciso di partire da un unico centro di tuple. Durante l'evoluzione dell'esperimento successivamente a runtime verranno creati dinamicamente altri centri che rappresenteranno i vari cluster. L'obbiettivo finale è quello di avere un centro di tuple per ogni cluster di articoli "simili". Nel centro di tuple iniziale vengono inserite tutte tuple che rappresentano ogni articolo scientifico ognuna delle quali ha la seguente struttura:

article(nome file, parola chiave 1, parola chiave 2, ..., parola chiave n).

Un esempio di tupla inserita è il seguent:

article(paper2, proceedings, 13th, workshop, object, agent, ..., september).

Questa semplice mansione è svolta da un singolo agente istanza della classe denominata TupleCentreInitializer, che alla fine del suo processo farà partire l'agente che si occupera del clustering.

3.1.2 Algoritmo di clustering

L'algoritmo di clustering è interamente svolto da un singolo agente istanza della classe denominata OrganizationAgent. L'agente prende una tupla, inizialmente la prima tupla del centro di tuple appena inizializzato, ne calcola la congruenza rispetto alle altre. Se il valore della congruenza è superiore alla soglia di congruenza specificata non fa nulla e passa alla tupla successiva. Se invece è inferiore allora controlla la congruenza della tupla rispetto alle tuple di ogni altro centro di tuple esistente. Se non ne esistono altri oltre a quello di origine o se non ne esiste uno che che possa contenere la tupla crea un nuovo centro di tuple. Terminato il controllo la tupla viene rimossa dal suo centro di tuple di origine e viene inserito nel centro individuato al passo precedente. Il processo continua fino a che tutte le tuple del centro di tuple iniziale non sono state controllate.

3.2 Esperimenti con altre algoritmi di clustering

Vengono qui presentate gli altri due algoritmi di clustering realizzati all'interno del progetto. Il primo algoritmo si differenzia dall'algoritmo basilare per quel che riguarda la funzione di matching fra le tuple che rappresentano gli articoli. Il secondo ed ultimo è un'ulteriore evoluzione dell'algoritmo sopra oltre alla funzione di matching modificata, il numero di agenti che si occupano del clustering è molto più alto. Ad ogni tupla viene associato un agente che si occuperà della sua sistemazione nel centro di tuple (cluster) più indicato.

3.2.1 Clustering con matching a similarità del coseno

Questo algoritmo come sopra accennato ha un sistema di matching diverso rispetto al mero matching sintattico. Il matching viene fatto utilizzando i concetti e la formula della similarità del coseno. Innanzi tutto ogni articolo viene rappresentato come un vettore di pesi ognuo associato ad una singola parola. Si ottiene cosi uno spazio di vettori di numeri reali che verranno poi inseriti nella formula del calcolo dela similarità del coseno. Nel processo per calcolare la similarità del coseno e necessario redigere in primis un vocabolario che contiene al suo interno tutte le parole presenti nei vari testi. Tenendo traccia per ogni singola parola degli articoli che la contengono. Il peso di ogni singola parola per ogni singolo articolo viene calcolato secondo la seguente formula: I pesi sono stati calcolati sulla base di formula ad-hoc che combina risultato dalla funzione di Okapi

e l'esito normalizzato ottenuto calcolando il valore idf della INQUERY. Ogni peso viene calcolato in base alla seguente funzione:

$$v_{i,j} = \frac{tf_{i,j}\log(\frac{colsize+0.5}{docf_i})}{tf_{i,j}+0.5+1.5\frac{doclen_j}{avgdoclen}\log(colsize+1)}$$

dove $v_{i,j}$ è il peso dell'iesimo termine del vocabolario all'interno del documento jesimo, $tf_{i,j}$ è il numero di volte che il termine appare nel documento, $docf_i$ il numero di documenti in cui è presente il termine, $doclen_j$ è il numero di termini del documento, $avgdoclen$ è il numero medio di termini per i documenti della collezione. Se ad esempio una parola non è contenuta all'interno del testo il relativo peso (valore) sarà pari a zero. Si ottengono così dei vettori di lunghezza uguale per ogni articolo in esame. Quando vogliamo calcolare la similarità fra due vettori che rappresentano ognuno un articolo è sufficiente applicare la seguente equazione:

$$Sim(C_i, C_j) = \frac{\sum_{k=1}^{n} c_i(t_k)\cdot c_j(t_k)}{\sqrt{\sum_{k=1}^{n} c_i(t_k)^2 \sum_{k=1}^{n} c_j(t_k)^2}}.$$

Dove C_i e C_j sono i vettori dei pesi assegnati ai vari termini per il documento i-esimo e j-esimo. La parte restante dell'algoritmo rimane invariata tutto il lavoro è sempre svolto di organizzazione dei cluster è svolto da un singolo agente.

3.2.2 Clustering con calcolo distribuito

In questa implementazione, che si basa sul clustering con matching a similarità del coseno, prevede la distribuzione del carico di lavoro fra più agenti e non solo uno come negli algoritmi precedenti. L'algoritmo prevede che ad ogni tupla, inizialmente inserita nel centro di partenza, sia associato un agente istanza della Classe TupleOrganizationAgent. Si è reso per fare ciò necessario modificare inoltre l'agente che inizializza il sistema in modo tale da associare durante l'inserimento nel centro delle tuple un nuovo agente ad ogni tupla. Ogni singolo agente svolgerà la stessa mansione che faceva in precedenza l'unico agente, per fare ciò però dato che non è l'unico esistente nel sistema dovrà coordinarsi con gli altri agenti. Per fare ciò sfrutta un centro di tuple per coordinarsi con gli altri grazie all'immissione e rimozione di tuple dallo stesso. Prima di controllare la congruenza con altri centri di tuple esistenti ad esempio l'agente preleva dal centro di tuple di coordinazione una tupla del tipo sem(token). Dato che di questa tupla è presente un solo esemplare tutti gli altri agenti che vogliono fare la medesima operazione devono attendere che l'agente reimmetta alla fine dei suoi calcoli la tupla nello spazio.

3.3 Risultati ottenuti

Qui di seguito vengono presentati i risultati in termini di cluster ottenuti di alcuni degli algoritmi sopra citati. Gli articoli vengono rappresentati qui solo dal loro titolo, ogni

gruppo di articoli cluster è separato dall'altro da uno spazio. Il primo risultato presentato è la composizione dei cluster ottenuta utilizzando l'algoritmo di clustering con matching a similarità del coseno con soglia di similarità pari a 0.10.

```
'Molecules of Knowledge: Self-Organisation in Knowledge-Intensive Environments'
'Molecules of Knowledge: A Novel Perspective over Knowledge Management'
'Self-Organising News Management: The Molecules of Knowledge Approach'

'Probabilistic Modular Embedding for Stochastic Coordinated Systems'
'Towards the Analysis & Prediction of Complex System Behaviour in SAPERE'
'Probabilistic Embedding: Experiments with Tuple-based Probabilistic Languages'

'Simulation of caspases apoptotic signalling pathway in a tuple space-based bioin

'Cross-Network Opportunistic Collection of Urgent Data in Wireless Sensor Network
'Self-organising Semantic Resource Discovery for Pervasive Systems'
'A Simulation Framework for Pervasive Services Ecosystems'
'Gradient-based Self-organisation Patterns of Anticipative Adaptation'
'Composing gradients for a context-aware navigation of users in a smart-city'
'Combining self-organisation, context-awareness and semantic reasoning: the case

'Self-adaptive software needs quantitative verification at runtime'
'Adaptive organizational changes in agent-oriented methodologies'
'Adaptable Multi-Agent Systems: The Case of the Gaia Methodology'
'Engineering Pervasive Multiagent Systems in SAPERE'
'Stability Assessment of Aspect-Oriented Software Architectures: A Quantitative S
'Agent-based Conference Management: A Case Study in SODA'
'Processes Engineering & AOSE'
'Software Engineering for Self-Organizing Systems'
'Agents, Multi-Agent Systems and Declarative Programming: Who, What, When, Where,
'Building an Agent Methodology from Fragments: the MEnSA experience'

'Interdependent Artificial Institutions in Agent Environments'

'Description and Composition of Bio-Inspired Design Patterns: the Gradient Case'
'Towards Situated Awareness in Urban Networks: A Bio-inspired Approach'
'Description and composition of bio-inspired design patterns: a complete overview
'Description and Composition of Bio-Inspired Design Patterns: the Gossip Case'
'BIO-CORE: Bio-inspired Self-organising Mechanisms Core'

'Operational Semantics of Proto'
```

'Engineering Confluent Computational Fields: from Functions to Rewrite Rules'
'Chemical-Inspired Self-Composition of Competing Services'
'Linda in space-time: an adaptive coordination model for mobile ad-hoc environmen
'A Framework for Modelling and Simulating Networks of Cells'
'Core Operational Semantics of Proto'
'An Agent-based Model for the Pattern Formation in Drosophila Melanogaster'

'Implicit: A Multi-agent Recommendation System for Web Search'
'Situation Identification Techniques in Pervasive Computing: A Review'
'Deep diving into BitTorrent locality'

'Programming abstractions for integrating autonomous and reactive behaviors: an a
'Designing a general-purpose programming language based on agent-oriented abstrac
'From Actors and Concurrent Objects to Agent-Oriented Programming in simpal'
'Environment Programming in Multi-Agent Systems -- An Artifact-Based Perspective'
'Multi-agent Oriented Programming with JaCaMo'
'JaCa-Android: an agent-based platform for building smart mobile applications'
'Formalising the Environment in MAS Programming: A Formal Model for Artifact-Base
'Typing Multi-Agent Programs in simpAL'
'simpA: An Agent-oriented Approach for Programming Concurrent Applications on top
'Exploiting Agent-Oriented Programming for Building Advanced Web 2.0 Applications

'Coordination Models and Languages: From Parallel Computing To Self-Organisation'
'Nature-inspired Coordination for Complex Distributed Systems'
'Dynamic Composition of Coordination Abstractions for Pervasive Systems: The Case
'Nature-inspired Coordination Models: Current Status, Future Trends'
'Agents Writing on Walls: Cognitive Stigmergy and Beyond'

'Using SOA Governance Design Methodologies to Augment Enterprise Service Descript

'Sustainable Biomass Power Plant Location in the Italian Emilia-Romagna region'
'Debt Deleveraging and Business Cycles. An Agent-Based Perspective'

'Reaction Factoring and Bipartite Update Graphs
Accelerate the Gillespie Algorithm for Large-Scale
Biochemical Systems'

'Promoting Space-Aware Coordination: ReSpecT as a Spatial-Computing Virtual Machi

'Collaborative Learning and ICT: A Prototypal Learning Environment'

'Infrastructures and Tools for Multiagent Systems for the New Generation of Distr

'Towards Temporal Verification of Emergent Behaviours in Swarm Robotic Systems'
'Verifying the Evolution of Probability Distributions Governed by a DTMC'

'Unsupervised Learning of True Ranking Estimators using the Belief Function Frame

'Simulation and Analysis of Distributed Systems in Klaim'
'Toward Approximate Stochastic Model Checking of Computational Fields for Pervasi

'Towards a Coordination Approach to Adaptive Pervasive Service Ecosystems'
'Self-aware Pervasive Service Ecosystems'

'Towards a Logic Framework for Web Programming'

'Risk Analysis and Deployment Security Issues in a Multi-Agent System'

'Exploiting the JaCaMo framework for realising an adaptive room governance applic

'From Space to Stage: How Interactive Screens Will Change Urban Life'

'Description Spaces with Fuzziness'
'Coordination in Open and Dynamic Environments with TuCSoN Semantic Tuple Centres
'Semantic Tuple Centres'
'Coordinating e-Health Systems with TuCSoN Semantic Tuple Centres'

'BaSi: Multi-Agent Based Simulation for Medieval Battles'
'HomeManager: Testing Agent-Oriented Software Engineering in Home Intelligence'
'Simulation in Agent-Oriented Software Engineering: The SODA'

'Spatial Coordination of Pervasive Services through Chemical-inspired Tuple Space
'A Coordination Approach to Adaptive Pervasive Service Ecosystems'
'A Computational Framework for Multilevel Morphologies'
'On the Space-time Situation of Pervasive Service Ecosystems'
'Towards a comprehensive approach to spontaneous self-composition in pervasive ec
'Injecting Self-organisation into Pervasive Service Ecosystems'
'Self-organising Pervasive Ecosystems: A Crowd Evacuation Example'
'On Competitive Self-composition in Pervasive Services'
'A Survey on Nature-inspired Metaphors for Pervasive Service Ecosystems'
'A Model for Drosophila Melanogaster Development from a Single Cell to Stripe Pat
'Pervasive Ecosystems: a Coordination Model based on Semantic Chemistry'

```
'From SOA to Pervasive Service Ecosystems: An Approach based on Semantic Web tech
'A Coordination Approach to Spatially-Situated Pervasive Service Ecosystems'
'A Chemical Inspired Simulation Framework for Pervasive Services Ecosystems'
```

Si può notare già a occhio nudo che articoli che dal titolo si presume siano simili si trovino nello stesso cluster. Un'altro buon risultato è stato ottenuto anche sfruttando l'algoritmo più semplice di tutti quello con semplice matching sintattico delle parole con soglia di similarità pari a 0.40.

```
'Molecules of Knowledge: Self-Organisation in Knowledge-Intensive Environments'
'Molecules of Knowledge: A Novel Perspective over Knowledge Management'
'Self-Organising News Management: The Molecules of Knowledge Approach'

'Probabilistic Modular Embedding for Stochastic Coordinated Systems'
'Probabilistic Embedding: Experiments with Tuple-based Probabilistic Languages'

'Simulation of caspases apoptotic signalling pathway in a tuple space-based bioin

'Towards the Analysis & Prediction of Complex System Behaviour in SAPERE'

'Cross-Network Opportunistic Collection of Urgent Data in Wireless Sensor Network

'Self-adaptive software needs quantitative verification at runtime'

'Interdependent Artificial Institutions in Agent Environments'

'Self-organising Semantic Resource Discovery for Pervasive Systems'
'On the Space-time Situation of Pervasive Service Ecosystems'
'Composing gradients for a context-aware navigation of users in a smart-city'
'Towards Situated Awareness in Urban Networks: A Bio-inspired Approach'
'Towards a comprehensive approach to spontaneous self-composition in pervasive ec
'Combining self-organisation, context-awareness and semantic reasoning: the case
'Injecting Self-organisation into Pervasive Service Ecosystems'
'Pervasive Ecosystems: a Coordination Model based on Semantic Chemistry'

'Description and Composition of Bio-Inspired Design Patterns: the Gradient Case'
'Description and composition of bio-inspired design patterns: a complete overview
'Description and Composition of Bio-Inspired Design Patterns: the Gossip Case'
'BIO-CORE: Bio-inspired Self-organising Mechanisms Core'

'Operational Semantics of Proto'
```

'Core Operational Semantics of Proto'

'Implicit: A Multi-agent Recommendation System for Web Search'

'Adaptive organizational changes in agent-oriented methodologies'
'Adaptable Multi-Agent Systems: The Case of the Gaia Methodology'
'Agent-based Conference Management: A Case Study in SODA'
'BaSi: Multi-Agent Based Simulation for Medieval Battles'
'HomeManager: Testing Agent-Oriented Software Engineering in Home Intelligence'

'Programming abstractions for integrating autonomous and reactive behaviors: an a
'Designing a general-purpose programming language based on agent-oriented abstrac
'From Actors and Concurrent Objects to Agent-Oriented Programming in simpal'
'Environment Programming in Multi-Agent Systems -- An Artifact-Based Perspective'
'Multi-agent Oriented Programming with JaCaMo'
'JaCa-Android: an agent-based platform for building smart mobile applications'
'Typing Multi-Agent Programs in simpAL'
'simpA: An Agent-oriented Approach for Programming Concurrent Applications on top
'Exploiting Agent-Oriented Programming for Building Advanced Web 2.0 Applications

'Spatial Coordination of Pervasive Services through Chemical-inspired Tuple Space
'A Coordination Approach to Adaptive Pervasive Service Ecosystems'
'A Computational Framework for Multilevel Morphologies'
'Towards a Coordination Approach to Adaptive Pervasive Service Ecosystems'

'Coordination Models and Languages: From Parallel Computing To Self-Organisation'
'Nature-inspired Coordination for Complex Distributed Systems'
'Nature-inspired Coordination Models: Current Status, Future Trends'
'Agents Writing on Walls: Cognitive Stigmergy and Beyond'

'Using SOA Governance Design Methodologies to Augment Enterprise Service Descript

'A Simulation Framework for Pervasive Services Ecosystems'
'Self-organising Pervasive Ecosystems: A Crowd Evacuation Example'

'Sustainable Biomass Power Plant Location in the Italian Emilia-Romagna region'

'Reaction Factoring and Bipartite Update Graphs
Accelerate the Gillespie Algorithm for Large-Scale
Biochemical Systems'

'Gradient-based Self-organisation Patterns of Anticipative Adaptation'

'Promoting Space-Aware Coordination: ReSpecT as a Spatial-Computing Virtual Machi

'Collaborative Learning and ICT: A Prototypal Learning Environment'

'Engineering Confluent Computational Fields: from Functions to Rewrite Rules'
'Linda in space-time: an adaptive coordination model for mobile ad-hoc environmen

'Chemical-Inspired Self-Composition of Competing Services'
'On Competitive Self-composition in Pervasive Services'

'Infrastructures and Tools for Multiagent Systems for the New Generation of Distr

'Engineering Pervasive Multiagent Systems in SAPERE'

'A Framework for Modelling and Simulating Networks of Cells'
'A Model for Drosophila Melanogaster Development from a Single Cell to Stripe Pat

'Dynamic Composition of Coordination Abstractions for Pervasive Systems: The Case

'Towards Temporal Verification of Emergent Behaviours in Swarm Robotic Systems'

'Unsupervised Learning of True Ranking Estimators using the Belief Function Frame

'Formalising the Environment in MAS Programming: A Formal Model for Artifact-Base

'Situation Identification Techniques in Pervasive Computing: A Review'

'Stability Assessment of Aspect-Oriented Software Architectures: A Quantitative S

'Simulation and Analysis of Distributed Systems in Klaim'

'Towards a Logic Framework for Web Programming'

'Toward Approximate Stochastic Model Checking of Computational Fields for Pervasi

'Risk Analysis and Deployment Security Issues in a Multi-Agent System'

'Exploiting the JaCaMo framework for realising an adaptive room governance applic

```
'Processes Engineering & AOSE'
'Building an Agent Methodology from Fragments: the MEnSA experience'
'Simulation in Agent-Oriented Software Engineering: The SODA'

'From Space to Stage: How Interactive Screens Will Change Urban Life'

'A Survey on Nature-inspired Metaphors for Pervasive Service Ecosystems'
'From SOA to Pervasive Service Ecosystems: An Approach based on Semantic Web tech
'A Coordination Approach to Spatially-Situated Pervasive Service Ecosystems'

'Software Engineering for Self-Organizing Systems'

'Deep diving into BitTorrent locality'

'Description Spaces with Fuzziness'
'Coordination in Open and Dynamic Environments with TuCSoN Semantic Tuple Centres
'Semantic Tuple Centres'
'Coordinating e-Health Systems with TuCSoN Semantic Tuple Centres'

'An Agent-based Model for the Pattern Formation in Drosophila Melanogaster'

'Agents, Multi-Agent Systems and Declarative Programming: Who, What, When, Where,

'Verifying the Evolution of Probability Distributions Governed by a DTMC'

'Debt Deleveraging and Business Cycles. An Agent-Based Perspective'

'Self-aware Pervasive Service Ecosystems'

'A Chemical Inspired Simulation Framework for Pervasive Services Ecosystems'
```

I risultati differiscono in maniera evidente l'uno dall'altro ma si nota come in generale gli articoli che presumibilmente dovrebbero essere nello stesso cluster lo sono. La grande differenza nel valore della soglia di similarità non deve spaventare dato che nel caso del matching con similarità del coseno si tiene conto di molte più parole, l'intero insieme di parole contenute nei vari testi, e quindi i pesi sono di gran lunga più bassi.