EU-IST Project IST-2003-506826 SEKT
SEKT: Semantically Enabled Knowledge Technologies

D2.1.2 Ontology-Based Information Extraction (OBIE) v.2

Ting Wang (University of Sheffield)
Kalina Bontcheva (University of Sheffield)
Yaoyong Li (University of Sheffield)
Hamish Cunningham (University of Sheffield)

Abstract
EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D2.1.2 (WP2)
This deliverable presents an SVM-based approach for hierarchical relation extraction and experiments on ACE2004 training data. Automatic extraction of semantic relationships between instances in an ontology is necessary in order to attach richer semantic metadata to documents. We propose an SVM approach to hierarchical relation extraction, using features derived automatically from a number of GATE-based open-source language processing tools. We introduced several new, semantic features and investigated in detail the impact of various factors on performance: the features, the classification hierarchy and the amount of training data. Experimental results show a trade-off among these factors is important for the relation extraction task and as the relation hierarchy gets deeper, more semantic features are needed in order to improve the performance of the ontological relation extraction task.
Keyword list: Ontology-based Information Extraction, Machine Learning, Adaptive IE, Relation Extraction,

WP2 Metadata Generation
Prototype
Contractual date of delivery: 31/12/05
PU
Actual date of delivery: 18/01/06

CHANGES

Version	Date	Author	Changes
0.1	04.12.05	Ting Wang	Creation.
0.2	12.12.05	Ting Wang	Appendix added
0.3	12.12.05	Kalina Bontcheva	Minor edits
final	17.12.06	Kalina Bontcheva	Finalised according to reviewer comments

SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

British Telecommunications plc.
Orion 5/12, Adastral Park
Ipswich IP5 3RE
UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

Empolis GmbH
Europaallee 10
67657 Kaiserslautern
Germany
Tel: +49 631 303 5540
Fax: +49 631 303 5507
Contact person: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

Jozef Stefan Institute
Jamova 39
1000 Ljubljana
Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contact person: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

University of Karlsruhe, Institute AIFB
Englerstr. 28
D-76128 Karlsruhe
Germany
Tel: +49 721 608 6592
Fax: +49 721 608 6580
Contact person: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Tel: +44 114 222 1891
Fax: +44 114 222 1810
Contact person: Hamish Cunningham
E-mail: hamish@dcs.shef.ac.uk

University of Innsbruck
Institute of Computer Science
Techikerstraße 13
6020 Innsbruck
Austria
Tel: +43 512 507 6475
Fax: +43 512 507 9872
Contact person: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

Intelligent Software Components S.A.
Pedro de Valdivia, 10
28006
Madrid
Spain
Tel: +34 913 349 797
Fax: +49 34 913 349 799
Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

Kea-pro GmbH
Tal
6464 Springen
Switzerland
Tel: +41 41 879 00
Fax: 41 41 879 00 13
Contact person: Tom Bösser
E-mail: tb@keapro.net

Ontoprise GmbH
Amalienbadstr. 36
76227 Karlsruhe
Germany
Tel: +49 721 50980912
Fax: +49 721 50980911
Contact person: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

Sirma Group Corp., Ontotext Lab
135 Tsarigradsko Shose
Sofia 1784
Bulgaria
Tel: +359 2 9768 303, Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Vrije Universiteit Amsterdam (VUA)
Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contact person: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

Universitat Autonoma de Barcelona
Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vall` es)
Barcelona
Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contact person: Pompeu Casanovas Romeu
E-mail: pompeu.casanovas@uab.es

Siemens Business Services GmbH & Co. OHG
Otto-Hahn-Ring 6
81739 Munich
Germany
Contact person: Dirk Ramhorst
Tel: +49 (89)63640225; Fax: +49 89 63640233
Email: Dirk.Ramhorst@siemens.com

Executive Summary

This deliverable presents an SVM-based algorithm for hierarchical relation extraction and experiments on ACE2004 training data.

Automatic extraction of semantic relationships between instances in an ontology is necessary in order to attach richer semantic metadata to documents than is currently possible. In contrast to previous evaluations, ACE2004 introduced a type hierarchy for both entities (typically mapped to instances in an ontology) and relations (typically mapped to properties of instances), i.e., it is an important step towards ontology-based IE. We evaluate our method on the ACE2004 data, as it is a good starting point for experiments on learning ontological relations.

Previous work shows that using NLP (Natural Language Processing) tools to derive ML (Machine Learning) features can benefit the IE (Information Extraction) results. Various features which have been used for relation extractions include word, entity type, mention level, overlap, chunks, syntactic parse trees, and dependency, relations. Based on this previous work, by using a number of GATE-based open-source language processing tools, we developed a set of features for semantic relation extraction, among which several new, semantic features are introduced, including Part-Of-Speech, entity class, entity role, semantic representation of sentence and WordNet synonym set.

Experiments show that the performance improves as more linguistic input is provided to the learning algorithm, in the form of features. As the relation hierarchy gets deeper, more semantic features are needed in order to improve performance.

We also investigate in detail the impact of various factors on the performance: the features, the classification hierarchy and the amount of training data, which shows that a trade-off among these factors is important for the relation extraction task. In a nutshell, some kinds of linguistic information can be obtained more reliably than other (e.g., part-of-speech versus semantic representation of sentences), so the latter should only be used in cases where it leads to serious improvement in performance. Similarly, training data is expensive to obtain, so if the property (or relation) hierarchy is more shallow, less training data would be required.

Contents

1 Introduction

Information Extraction (IE) [1] is a process which takes unseen texts as input and produces fixed-format, unambiguous data as output. It involves processing text to identify selected information, such as particular named entity or relations among them from text documents. Named entities include people, organizations, locations and so on, while relations typically include physical relations (*located*, *near*, *part-whole*, etc.), personal or social relations (*business*, *family*, etc.), and membership (*employ-staff*, *member-of-group*, etc.).

Until recently, research has focused primarily on use of IE for populating ontologies with concept instances (e.g., [2] [3]). However, in addition to this, many ontology-based applications require methods for the automatic discovery of properties (called relations in the corpus used here) of instances. Semantic relations provide richer metadata connecting documents to ontologies and enable more sophisticated semantic search and knowledge access.

It is in this context that work on relation extraction becomes relevant. However, one of the main barriers to applying this work in ontology-based applications comes from the difficulty of adapting the algorithms to new domains. In order to overcome this problem, recent research has advocated the use of Machine Learning (ML) techniques for IE.

A number of ML approaches have been used for relation extraction, e.g. Hidden Markov Models (HMM) [4], Conditional Random Fields (CRF) [5] and Maximum Entropy Models [6]. Among those, Support Vector Machines (SVM) [7] have been particularly successful and tended to outperform other methods.

Zelenko et al. [10] proposed extracting relations by computing kernel functions between shallow parse trees [8]. Kernels have been defined over shallow parse representations of text and have been used in conjunction with SVM learning algorithms for extracting *person-affiliation* and *organization-location* relations. Culotta et al. [9] extended this work to estimate kernel functions between augmented dependency trees.

Zhou et al. [10] further introduced diverse lexical, syntactic and semantic knowledge in feature-based relation extraction using SVM. The feature system covers word, entity type, overlap, base phrase chunking, dependency tree and parse tree, together with relation-specific semantic resources, such as country name list, personal relative trigger word list. Their results show that the feature-based approach outperforms tree kernel-based approaches, achieving 55.5% F-measure [1] in relation detection and classification on the ACE2003 training data. This is in fact a state-of-the-art performance on this task, as it is a lot harder than other information extraction tasks, e.g., recognising names of people.

Motivated by this work, we apply a diverse set of Natural Language Processing (NLP)

[1] F-measure is a standard performance metrics used for evaluating Information Extraction systems against each other. It is a combination of two other frequently reported measures – precision and recall. For further information, see [20].

tools to derive features for relation extraction. In particular, several new, semantic features are introduced, including Part-Of-Speech, entity class, entity role, semantic representation of sentences and synonymous words.

In the rest of the deliverable, we first describe the ACE2004 entity and relation type hierarchy from an ontological perspective (Section 2). Then we give a brief introduction to SVMs used as the classifier for relation extraction (Section 3) and introduce GATE, which provides a set of NLP tools for deriving an extensive set of features (Section 4). Section 5 presents and discusses a series of experiments that investigate the impact of of various factors on performance. Finally, we summarise the work and discuss some future directions.

2 The ACE Entity and Relation Hierarchies

Relation extraction from text aims to detect and classify semantic relations between entities according to a predefined entity and relation hierarchy or an ontology. The Automatic Content Extraction (ACE) programme [11] defines this task as Relation Detection and Characterization (RDC). RDC uses the results of named entity recognition, which detects and classifies entities according to a predefined entity type system.

In contrast to previous evaluations, ACE2004 introduced a type and subtype hierarchy for both entities and relations, i.e., it is an important step towards ontology-based IE. We evaluate our method on the ACE2004 data, as it is a good starting point for experiments on learning ontological relations.

2.1 The ACE2004 Entity Hierarchy

Entities are categorized in a two level hierarchy, consisting of 7 types and 44 subtypes as shown in Table 1 [12].

Table 1. ACE2004 entity types and subtypes.

Type	Subtype
Person (PER)	*(none)*
Organization (ORG)	*Government, Commercial, Educational, Non-profit, Other*
Facility (FAC)	*Plant, Building, Subarea_Building, Bounded Area, Conduit, Path, Barrier, Other*
Location (LOC)	*Address, Boundary, Celestial, Water_Body, Land_Region_natural, Region_Local, Region_Subnational, Region_National, Region_International, Other*
Geo-Political Entities (GPE)	*Continent, Nation, State/Province, County/District, City/Town, Other*
Vehicle (WEH)	*Air, Land, Water, Subarea_Vehicle, Other*
Weapon (WEA)	*Blunt, Exploding, Sharp, Chemical, Biological, Shooting, Projectile, Nuclear, Other*

Each entity has been assigned a class which describes the kind of reference the entity makes to something in the world. The class can be one of four values [12]:

- *Negatively Quantified (NEG):* the entity refers to the empty set of the type of

entity mentioned, i.e., indicates that no such entity exists, e.g., [No one] can guarantee the success.

- *Specific Referential (SPC):* the entity being referred to is a particular, unique object (or set of objects), e.g. [Mike's friend] appreciated what he has done.
- *Generic Referential (GEN):* the entity being referred to is not a particular, unique object (or set of objects), e.g., [Friends] are lost by calling often...
- *Under-specified Referential (USP):* indicates non-generic non-specific reference, for example: I don't know [how many people] came.

The occurrence of each entity in the dataset is called an *entity mention*, which can be one of the following [12]:

- *Names (NAM):* indicates that the entity mention is a proper noun and nickname, such as: [Chancellor Gordon Brown] has announced the new plan.
- *Quantified Nominal Constructions (NOM):* the entity mention is a noun quantified with a determiner, a quantifier, or a possessive, e.g.: John stands on [the top of the mountain].
- *Pronouns (PRO):* the entity mention is a pronoun with the exception of wh-question words and the specifier 'that', e.g., *he, I, they.*
- *Pre-modifier (PRE):* mentions, which occur in a modifying position before another word(s), e.g. Best is one of the greatest [British] football players.

In addition, geo-political entities are regarded as composite entities comprising of population, government, physical location, and nation (or province, state, county, city, etc.). Consequently, each GPE mention in the text has a *mention role* which indicates which of these four aspects is being referred to in the given context: *Person (PER)*, *Organization (ORG)*, *Location (LOC)*, and *GPE*.

2.2 The ACE2004 Relation Hierarchy

In an ontology, the concepts are not only organised in a taxonomy representing *IS-A* relations, but also linked together by semantic relations such as *Part-Whole*, *Subsidiary*, *LocatedIn*, etc. ACE2004 defines a hierarchy of relations with 7 top types and 22 sub-types, shown in Table 2 [13]:

Table 2. ACE2004 relation types and subtypes.

Type	Subtype
Physical (PHYS)	*Located, Near*, Part-Whole*
Personal/Social (PER-SOC)	*Business*, Family*, Other**
Employment/Membership/ Subsidiary (EMP-ORG)	*Employ-Exec, Employ-Staff, Employ-Undetermined, Member-of-Group, Subsidiary, Partner*, Other**
Agent-Artifact (ART)	*User/Owner, Inventor/Manufacturer, Other*
PER/ORG Affiliation (OTHER-AFF)	*Ethnic, Ideology, Other*
GPE Affiliation (GPE-AFF)	*Citizen/Resident, Based-In, Other*
Discourse (DISC)	*(none)*

This relation type and subtype hierarchy can also be depicted as a three levels tree (see Fig 1).

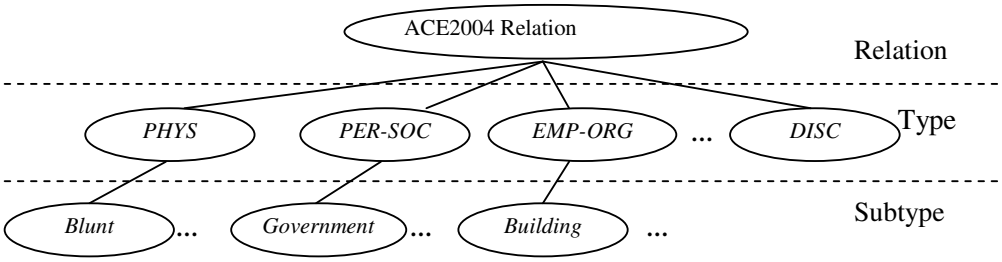

Fig. 1. The hierarchy of the ACE2004 relation types and subtypes

There are 6 symmetric relations (marked with a star in the table 2) and the remaining ones are asymmetric relations[2]. Below are some examples:

Table 3. Examples of ACE2004 relations.

Examples	Realtions -- Type.Subtype(*arg1*, *arg2*)
the president of US	EMP-ORG.Employ-exec(*president*, *US*)
some Missouri voters	GPE-AFF.Citizen/Resident(*voters*, *missouri*)
part of Louisiana	PHYS. Part-Whole(*part*, *Louisiana*)
John's superiors	PER-SOC.Business(*John*, *superiors*)
Fox comes from a rich family.	PER-SOC.Family(*Fox*, *family*)
Jean's computer has crashed.	ART.User-Owner(*Jean*, *computer*)

In the experiments reported next, we use the ACE2004 corpus to evaluate ontological relation extraction. The ACE corpus consists of the following types of input data: text from newswire; broadcast news; and newspapers. The texts are unstructured and encompass a wide variety of domains, such as sport, politics, religion, popular culture, etc.

3 Using SVM for Relation Extraction

SVM is one of the most successful ML methods for IE, which has achieved the state-of-the-art performance on many classification and IE tasks. As SVM were originally designed for binary classification, much work has been done to extend it for multi-class classification, which can be divided into two types [14]: *(i)* constructing and combining several binary classifiers; *(ii)* directly considering all data in one optimization formulation.

This work uses the LIBSVM implementation of an SVM classifier for multi-class classification [14]. For a *k*-class classification task, this method constructs $k(k-1)/2$ classifiers where each one is trained on data from two classes. LIBSVM uses the "Max Wins" voting strategy to predict the class [14]: apply every classifier to predict x; if one classifier says x is in the *i*th class, then the vote for the *i*th class is incremented by one; in the end x is classified in the class with the highest number of

[2] Symmetry of relation is important when determining which entity is the subject and which is the object of the relation.

votes.

We built SVM models for detecting the relations, predicting the type and subtype of relations between every pair of entity mentions within the same sentence. As defined in the ACE evaluation, we only model explicit relations rather than implicit ones[3]. For example, the sentence

Texas has many cars. (1)

explicitly expresses a *ART.User/Owner* relation between the two entity mentions *Texas* and *many cars*. The relation extraction system needs to determine the relation and its type and subtype based on the context information within this sentence. The procedure used to do that is described in the following section. However, end-users and readers not interested in the technical details can skip this section and continue with Section 5.

4 Features for Relation Extraction

Using NLP tools to derive ML features has been shown to benefit IE results. Features which have been used for relation extraction include word, entity type, mention level, overlap, chunks, syntactic parse trees, and dependency relations [4] [6] [8] [10].

Based on this previous work, we developed a set of features for semantic relation extraction, many of which are adopted from [10]. Further, we introduce some new features such as Part-Of-Speech (POS) tags, entity subtype and class features, entity mention role feature, and several general semantic features. Zhou et al. [10] have designed some relation-specific semantic features. For example, some important trigger word lists have been collected from WordNet [15] in order to differentiate the six personal social relation subtypes. However, these lists are too specific to the dataset to be applicable for general purpose relation extraction. Therefore in our method, we introduce instead a set of more general semantic features produced by a semantic analyser and WordNet.

WordNet [15] is a widely used linguistic resource which is designed according to psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets (called *synsets*), each representing one underlying lexical concept. In this work, WordNet is used to derive several semantic features based on the synset and hypernym information.

NP (Noun Phrase) and VP (Verb Phrase) Chunkers are NLP modules which identify the noun and verb phrases in each sentence.

BuChart (which has been renamed to SUPPLE) is a bottom-up parser that constructs syntax trees and logical forms for English sentences [16]. One of its significant characteristics is that it can produce a semantic representation of sentences – called simplified quasi-logical form (SQLF). Previously, one of the limitations in applying general semantic information in IE is the relative lack of robustness of semantic analysers. However, BuChart is a general purpose parser that can still produce partial

[3] Explicit relations have surface linguistic forms that signal them, e.g., possessive, appositions, prepositional phrases. Implicit relations are relations that can be inferred from the text, but are not clearly signalled, e.g., a text mentioning "Prime minister Tony Blair" and "Britain" implicitly conveys a relation that Tony Blair has a job as prime minister of Britain. For further details see [10].

syntactic and semantic results for fragments even when the full sentential parses cannot be determined. Therefore, it is particularly suitable for deriving semantic features for ML-based relation extraction , because real-world texts frequently contain sentences that are difficult to parse in full.

4.1 Using GATE for Feature Extraction

General Architecture for Text Engineering (GATE) [17] is an infrastructure for developing and deploying software components that process human language. It provides a set of NLP tools including a tokeniser, gazetteer, POS tagger, chunker, parsers, etc. For the relation extraction task, we make use of a number of GATE components as follows:

- English Tokeniser
- Sentence Splitter
- POS Tagger
- NP Chunker
- VP Chunker
- BuChart Parser
- MiniPar Parser
- WordNet

4.2 Word Features

This set consists of 14 features including the word list of the two entity mentions and their heads, the two words before the first mention, the two after the second mention, and the word list between them. In the following, we use M1 and M2 denote the first and second entity mention involved in the relation, H1 and H2 denote the heads of the entity mentions. The word level features are defined as:

- WM1: word list of M1
- HM 1: the head word of M1
- BM1F: the first word before M1
- BM1L: the second word before M1
- WM2: word list in M2
- HM2: head word of M2
- AM2F: the first word after M2
- AM2L: the second word after M2
- HM12: combination of HM1 and HM2
- WBNULL: if there is no word in between
- WBFL: the only word in between when only one word in between
- WBF: first word in between when at least two words in between
- WBL: last word in between when at least two words in between
- WBO: other words in between except first and last words when at least three words in between

For the example sentence (1), it contains two entity mentions as:

[Texas] has [many cars].

So the word features the two mentions are:

Table 4. The word features for the example sentence.

Feature	Value
WM1	Texas
HM1	Texas
BM1F	NULL
BM1L	NULL
WM2	many cars
HM2	cars
AM2F	.
AM2L	NULL
WBNULL	False
WBFL	has
WBF	NULL
WBL	NULL
WBO	NULL

4.3 POS Tag Features

Because the word features are often too sparse, we also introduce POS tag features.

Similar to the word features, this set of features includes the POS tag list of the two entity mentions and their heads, the two POS tags before the first mention, the two after the second mention, and the tag list in between. The POS tag level features are defined as:

- HM1POS: POS tag(list) of the head word(list) of M1
- BM1FPOS: POS tag of the first word before M1
- BM1LPOS: POS tag of the second word before M1
- HM2POS: POS tag(list) of the head word(list) of M2
- AM2FPOS: POS tag of the first word after M2
- AM2LPOS: POS tag of the second word after M2
- POS12: combination of POS tags of HM1 and HM2
- POSBFL: POS tag of the only word in between when only one word in between
- POSBF: POS tag of the first word in between when at least two words in between
- POSBL: POS tag of the last word in between when at least two words in between
- POSBO: POS tags of other words in between except first and last words when at least three words in between

For example, sentence (1) has been tagged as: Texas/NNP has/VBZ many/JJ cars/NNS, where NNP denotes proper name, JJ – adjectives, NNS – plural nouns, etc. The values of the corresponding features are shown in Table 5 below:

Table 5. The POS tag features for the example sentence.

Feature	Value
M1POS	NNP
HM1POS	NNP
BM1FPOS	NULL
BM1LPOS	NULL
M2POS	JJ NNS
HM2POS	NNS
AM2FPOS	.
AM2LPOS	NULL
POS12	False
POSBFL	has
POSBF	NULL
POSBL	NULL
POSBO	NULL

4.4 Entity Features

As already discussed, ACE2004 divides entities into seven types, as shown in Table 1. For each pair of mentions, the combination of their entity types is taken as the entity type feature. For the example sentence above, there are two entity mentions: *Texas* categorized as *GPE (Geo-Political Entities)*, and *many cars* which is *WEH (Vehicle).*

In addition, the entity hierarchy is used, because subtypes carry more accurate semantic information for the entity mentions. Therefore, the combination of the entity subtypes of the two entity mentions is provided as the entity subtype feature. The subtypes of the two example mentions are *State-or-Province* and *Land.*

Finally, each entity has also been annotated with a class which describes the kind of reference for the entity. So the entity class is also used in this paper to predicate semantic relations. The classes for the above two mentions are *SPC (Specific Referential)* and *USP (Under-Specified Referential).*

The definitions of these features are listed as follows:

- ET12: combination of the mention entity types of both mentions
- EST12: combination of the mention entity subtypes of both mentions
- EC12: combination of the mention entity classes of both mentions

The values of these features are:

Table 6. The entity features for the example sentence.

Feature	Value
ET12	GPE + WEH
EST12	State-or-Province + Land
EC12	SPC + USP

4.5 Mention Features

This set of features includes the mention type and role information of both mentions. For the example sentence, the mention types for the two mentions in sentence (1) are *NAM (Name)* and *NOM (Nominal),* while the mention role for *Texas* is *GPE* and there is no role information for the second because only a GPE can take role information. The definitions are:

- MT12: combination of the mention types of both mentions
- MR12: combination of the mention roles of both mentions
- MM12: combination of the mention metonymies of both mentions

The entity features for the example sentence are:

Table 7. The mention features for the example sentence.

Feature	Value
MT12	NAM + NOM
MR12	GPE + NULL
MM12	NULL +NULL

4.6 Overlap Features

The relative position of the two entity mentions can also be helpful for indicating the relationship between them. For these features, we have considered: the number of words separating them, the number of other entity mentions in between, whether one mention contains the other. As the feature indicating whether one mention contains the other is too general, it has been combined with the entity type and subtype of the two mentions to form more discriminating features. The definitions are:

- NUMMB: number of other mentions in between
- NUMWB: number of words in between
- CTN12: flag indicating whether M2 is included in M1.
- CTN21: flag indicating whether M1is included in M2.
- ET12 CTN12: combination of ET12 and CTN12
- ET12 + CTN21: combination of ET12 and CTN21
- EST12 + CTN12: combination of EST12 and CTN12
- EST12 + CTN21: combination of EST12 and CTN21

For the example sentence, these values are:

Table 8. The overlap features for the example sentence.

Feature	Value
NUMMB	0
NUMWB	1
M1CNTM2	false
M2CNTM1	false
ET12M1CNTM2	GPE + WEH + false
ET12M2CNTM1	GPE + WEH + false
EST12M1CNTM2	State-or-Province + Land + false
EST12M2CNTM1	State-or-Province + Land + false

4.7 Chunk Features

GATE integrates two chunk parsers: Noun Phrase (NP) and Verb Phrase (VP) Chunker that segment sentences into noun and verb group chunks. For instance, the example sentence (1) is chunked as: [*Texas*] {has} [*many* cars], in which, *Texas* and *many cars* are NPs, while *has* is the VP between them whose voice and type are respectively *active and FVG* (abbreviation for Finite Verb Group, also referred to verb phrase by some other NLP components).

The following information has been used as chunk features: whether the two entity mentions are included in the same NP chunk or VP chunk, the type and voice information of the verb group chunk in between if there is any.

The definition of the chunk features are listed as:

- SAMENP: whether M1 and M2 included in the same Noun Phrase
- SAMEVP: whether M1 and M2 included in the same Verb Phrase
- VPB: type + voice of the verb group chunk in between

The values of these features for the example sentence are:

Table 9. The chunk features for the example sentence.

Feature	Value
SAMENP	false
SAMEVP	false
VPB	FVG + active

4.8 Dependency Features

In contrast to Kambhatla [6] and Zhou et al. [10], who derive the dependency tree from the syntactic parse tree, we apply MiniPar to directly build the dependency tree. MiniPar is a shallow parser which can determine the dependency relationships between the words of a sentence [18]. Fig 2 shows the dependency tree for the example sentence.

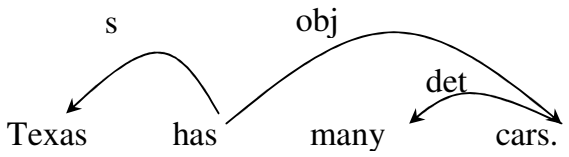

Fig. 2. The dependency tree for the example sentence.

From the resulting dependency relationships between words, the dependency features are formed, including: combination of the head words and their dependent words for the two entity mentions involved; the combination of the dependency relation type and the dependent word of the heads of the two mentions; the combination of the entity type and the dependent word for each entity mention's head; the name of the dependency relationship between the heads of the two mentions if there is any; the word on which both the heads of the two mentions depend on if there is any; and the path of dependency relationship labels connecting the heads of the two mentions.

The detailed definitions of these dependency features are:

- ET1DW1: combination of the entity type and the dependent word for M1
- H1DW1: combination of the head word and the dependent word for M1
- ET2DW2: combination of the entity type and the dependent word for M2
- H2DW2: combination of the head word and the dependent word for M2
- DT1DW1: combination of the dependency relation type and the dependent word of the head word of M1
- DT2DW2: combination of the dependency relation type and the dependent word of the head word of M2
- DT12: the name of the dependency relationship between the heads of the M1 and M2
- DT21: the name of the dependency relationship between the heads of the M2 and M1
- SAMEDW: if the heads of the two mentions depend on same word
- DW12: the word on which both the heads of the two mentions depend on
- DRP: path of dependency relationship labels connecting the heads of two mentions

For the example sentence, its dependency features are:

Table 10. The dependency features for the example sentence.

Feature	Value
ET1DW1	GPE + has
H1DW1	Texas + has
DT1DW1	s + has
ET2DW2	VEH + has
H2DW2	cars + has
DT2DW2	obj + has
DT12	NULL
DT21	NULL
SAMEDW	TRUE
DW12	has
DRP	<s>obj

note: < and > denote the direction of dependency.

4.9 Parse Tree Features

The features on syntactic level are extracted from the parse tree. As we mentioned above, we use BuChart to generate the parse tree and the semantic representation of each sentence. Unlike many full parsers which would fail if a full sentential parse cannot be found, BuChart can still produce the partial parsing trees and corresponding semantic representations for the fragments.

The following lists the parse tree in the bracket form for the example sentence:

```
(s(np(bnp(bnp_core(bnp_head(ne_np(tagged_location_np(list_np"Texas")))))))
  (fvp(vp(vpcore(fvpcore(nonmodal_vpcore(nonmo-dal_vpcore1(vpcore1(av(v"has")))))))
        (np(bnp(bnp_core(premods(premod(jj"many")))
                         (bnp_head(n"cars")))))))
```

Before giving the features, we first introduce a term named *corresponding node* which is defined as: for a word sequence in the sentence parsed, say $W=w_i w_{i+1...} w_j$, if node *N* in the parse tree satisfied all of the following conditions:

1) *N* totally covers *W*;
2) if *N* has father *M*, *M*'s coverage is strictly bigger than *W*;
3) there is no such descendant of *N* who can totally cover *W* but its coverage is strictly smaller than the coverage of *N*.

We called N the *corresponding node* for the word sequence *W*.

We believe using such a corresponding node for phrase when designing features can achieve a balance between the over-specific and over-general. For examples, the corresponding node for *cars* is `bnp_head` (head of noun phrase), while the *corresponding* node for *many cars* is `np` (noun phrase), and the one for *Texas has many cars* is `s` (sentence).

Consequently, from the product of the parser, we extract the following features: the labels of the corresponding nodes and their fathers for each entity mentions involved; the labels of the corresponding node covering both entity mentions; the label of the corresponding node covering the heads of both entity mentions; the label of the path connecting the corresponding nodes of both mentions in the parse tree; and that connecting the heads of both mentions.

Here is the definition of these features:

- M1FPT: the labels of the corresponding node for M1
- M1LPT: the labels of the node who is the father of the corresponding node for M1
- M2FPT: the labels of the corresponding node for M2
- M2LPT: the labels of the node who is the father of the corresponding node for M2
- PTPM12: the labels of the path connecting the corresponding nodes of M1 and M2 in the parse tree
- PTM12: the label of the corresponding node covering both M1 and M2
- PTH12: the label of the corresponding node covering both H1 and H2
- PTPH12: the labels of the path connecting the corresponding nodes of H1 and H2 in the parse tree

Table 11. The parse tree features for the example sentence.

Feature	Value
M1FPT	np
M1LPT	s
M2FPT	np
M2LPT	vp
PTPM12	<np<s>fvp>vp>np
PTM12	s
PTH12	s
PTPH12	<np<s>fvp>vp>np> bnp>bnp_core>bnp_head

4.10 Semantic Features from SQLF

Using relation- or domain-independent semantic features potentially makes the approach easier to adapt to new domains. BuChart provides semantic analysis to produce SQLF for each phrasal constituent. The logical form is composed of unary predicates that denote entities and events (e.g., *chase(e1), run(e2)*) and binary predicates for properties (e.g., *lsubj(e1,e2)*[4]). Constants (e.g., e1, e2) are used to represent entity and event identifiers (see [16] for further details).

The (somewhat simplified) semantic analysis of the example sentence in SQLF is:
location(e2), name(e2,'Texas'), have(e1), time(e1,present), aspect(e1,simple), voice(e1,active), lobj(e1,e3), car(e3), number(e3,plural), adj(e3,many), lsubj(e1,e2)
From the SQLFs, a set of semantic features is generated, one of which is the path of predicate labels connecting the heads of both mentions in the semantic SQLFs. This path may be too specific to be effective and cause data sparseness problem, so we also take some important predicate labels as separate features, such as the first, second, last and penultimate predicates labels in that path.

So these semantic features from SQLF are defined as:

[4] *Lsubj* stands for logical subject and l*obj* – for logical object.

- PRDP: path of predicates labels connecting the head of M1 and M2 in the semantic QLF
- FPRDP: the first predicate in PRDP
- SPRDP: the second predicate in PRDP
- LPRDP: the last predicate in PRDP
- SLPRDP: the second last predicate in PRDP
- PRDPFL: the only predicate in between when only one predicate in between

The values of Semantic Features from SQLF for the example sentence are in the following table:

Table 12. The semantic features from SQLF for the example sentence.

Feature	Value
PRDP	lsubj + lobj
FPRDP	lsubj
SPRDP	lobj
LPRDP	lobj
SLPRDP	lsubj
PRDPFL	NULL

4.11 Semantic Features from WordNet

To exploit more relation-independent semantic features, we use WordNet together with a simple semantic tagging method to find the sense information for the words in each sentence. Tagging words with their corresponding WordNet synsets (i.e., word sense disambiguation - WSD) is a difficult task, which usually can not achieve accuracy as high as other NLP tasks such as POS tagging. However, WordNet's design ensures that synsets are ordered by importance, so a simple and yet efficient heuristic can be used instead of a WSD module, without major accuracy penalty[5]. The heuristic is to take the first synset from WordNet, which matches the POS tag of the given word. Each synset has been assigned an id (consisting of the POS tag and its offset in the WordNet files) which is used in the features.

Similar to the word and POS tag features, the features from WordNet (called WordNet synset features) include the synset-id list of the two entity mentions and their heads, the two synset-ids before the first mention, the two after the second mention, and the synset-id list in between. With considerations of the data sparseness problem, we also developed a set of more abstract features by using the hypernym information of each synset (called WordNet hypernym features). The hypernym features parallel the synset ones by replacing each synset-id with the id of its hypernym synset.

The definition of the semantic features from WordNet are categorised into two set: one is Synset set, the other is Hypernym set. Here list the:

[5] The first-sense heuristic achieves precision and recall around 60%, whereas the best WSD methods – around 70% (see Figure 1 in [19]).

Synset set:

- M1SYNSET: Synset (List) of word(list) of M1
- HM1SYNSET: Synset (List) of head word(list) of M1
- BM1FSYNSET: Synset of first word before M1
- BM1LSYNSET: Synset of second word before M1
- M2SYNSET: Synset (List) of word(list) of M2
- HM2SYNSET: Synset (List) of head word(list) of M2
- AM2FSYNSET: Synset of first word after M2
- AM2LSYNSET: Synset of second word after M2
- SYNSET12: combination of SYNSETs of the H1 and H2
- SYNSETBFL: Synset of the only word in between when only one word in between
- SYNSETBF: Synset of first word in between when at least two words in between
- SYNSETBL: Synset of last word in between when at least two words in between
- SYNSETBO: Synset s of other words in between except first and last words when at least three words in between

Hypernym set:

- M1SYNPARENT: Direct Hypernym Synset (List) of word(list) of M1
- HM1SYNPARENT: Direct Hypernym Synset (List) of head word(list) of M1
- BM1F SYNPARENT: Direct Hypernym Synset of first word before M1
- BM1L SYNPARENT: Direct Hypernym Synset of second word before M1
- M2SYNPARENT: Direct Hypernym Synset (List) of word(list) of M2
- HM2SYNPARENT: Direct Hypernym Synset (List) of head word(list) of M2
- AM2FSYNPARENT: Direct Hypernym Synset of first word after M2
- AM2LSYNPARENT: Direct Hypernym Synset of second word after M2
- SYNPARENT12: combination of Direct Hypernym SYNPARENT s of the HM1 and HM2
- SYNPARENTBFL: Direct Hypernym Synset of the only word in between when only one word in between
- SYNPARENTBF: Direct Hypernym Synset of first word in between when at least two words in between
- SYNPARENTBL: Direct Hypernym Synset of last word in between when at least two words in between
- SYNPARENTBO: Direct Hypernym Synset s of other words in between except first and last words when at least three words in between

Table 13 list the values of these features for the example sentence.

Table 13. The semantic features from WordNet for the example sentence.

Feature	Value	Feature	Value
M1SYNSET	noun_8574816	M1SYNPARENT	noun_8126624
HM1SYNSET	noun_8574816	HM1SYNPARENT	noun_8126624
BM1FSYNSET	NULL	BM1FSYNPARENT	NULL
BM1LSYNSET	NULL	BM1LSYNPARENT	NULL
M2SYNSET	adjective_1500322 noun_2853224	M2SYNPARENT	NULL + noun_3649150
HM2SYNSET	noun_2853224	HM2SYNPARENT	noun_3649150
AM2FSYNSET	NULL	AM2FSYNPARENT	NULL
AM2LSYNSET	NULL	AM2LSYNPARENT	NULL
SYNSET12	noun_8574816+ noun_2853224	SYNPARENT12	noun_8126624 + noun_3649150
SYNSETBFL	verb_2139918	SYNPARENTBFL	NULL

SYNSETBF	NULL	SYNPARENTBF	NULL
SYNSETBL	NULL	SYNPARENTBL	NULL
SYNSETBO	NULL	SYNPARENTBO	NULL

5 Experimental Results

We evaluate our method, especially the contribution of the different features, on the ACE2004 training data. As mentioned above in Section 3, only explicit relations between pairs of entity mentions within the same sentence are considered. We not only evaluate the performance of the system as a whole, but furthermore, we also investigate in detail several factors which have impact on the performance, such as the features set and the relation classification hierarchy.

5.1 Experimental Settings

The ACE2004 training data consists of 451 annotated files (157,953 words) from broadcast, newswire, English translations of Arabic and Chinese Treebank, and Fisher Telephone Speech collection. Among these files, there are 5,914 relation instances annotated which satisfied the experiment set up described above. The data has been divided into two sets: the training set consists of 361 files, 4,744 relation instances; the testing set consists of 90 files which contains 1,170 instances. The distribution of the instances is listed in Table 14.

Table 14. The distributions of the relation instances in ACE 2004 training data.

Type	Subtype	Number		
		Total	Train	Test
PHYS		**1,688**	**1,331**	**357**
	Located	**1,029**	**809**	**220**
	Near	**141**	**106**	**35**
	Part-Whole	**518**	**416**	**102**
PER-SOC		**444**	**366**	**78**
	Business	**197**	**163**	**34**
	Family	**178**	**143**	**35**
	Other	**69**	**60**	**9**
EMP-ORG		**2,084**	**1,675**	**409**
	Employ-Exec	**630**	**502**	**128**
	Employ-Staff	**694**	**562**	**132**
	Employ-Undetermined	**129**	**94**	**35**
	Member-of-Group	**225**	**185**	**40**
	Subsidiary	**300**	**238**	**62**
	Partner	**16**	**13**	**3**
	Other	**90**	**81**	**9**
ART		**293**	**233**	**60**
	User/Owner	**273**	**222**	**51**
	Inventor/Manufacturer	**13**	**8**	**5**
	Other	**7**	**3**	**4**
OTHER-AFF		**183**	**143**	**40**
	Ethnic	**53**	**43**	**10**
	Ideology	**55**	**42**	**13**
	Other	**75**	**58**	**17**
GPE-AFF		**788**	**650**	**138**
	Citizen/Resident	**368**	**289**	**79**
	Based-In	**333**	**284**	**49**
	Other	**87**	**77**	**10**
DISC		**434**	**346**	**88**

Total		**5,914**	**4,744**	**1,170**

Following previous work, in order to focus on the performance of the relation extraction only, we suppose that all named entities have been recognised without mistakes and only evaluate the performance of relation extraction on "true" named entity mentions with correct co-reference chaining between them (i.e. as annotated by the human annotators).

Among the 23 relation subtypes (including *DISC* which has no subtype), there are 6 symmetric ones. So to model the relation extraction task as multi-class classification, we use two labels to denote each non-symmetric relation and only one label for each symmetric one. Also we assign a label to the class of *no-relation*, which indicates that there is no relation between the two entity mentions. Consequently, in our experiments, relation extraction is modelled as a 41-class classification task, where each pair of entity mentions is assigned one of these 41 relation classes, based on the features discussed in Section 4. In the following experiments, we use LIBSVM as the SVM classifier with a linear kernel function and one-against-one method for multi-class classification which has been described in Section 3.

From Table 1 we can see that the different relation subtypes and types are distributed very unevenly, so we only measure the micro-average of Precision, Recall and F1, because in such cases macro-average does not reflect the performance reliably.

5.2 Experiments on Features

The first experiments investigate the impact of different features on the performance by adding them incrementally. The results are presented in Table 15.

Table 15. The result on different feature sets.

Features	Precision	Recall	F1
Word	**55.19**	**25.47**	**34.85**
+POS Tag	**59.61**	**28.63**	**38.68**
+Entity	**61.32**	**45.38**	**52.16**
+Mention	**62.96**	**46.92**	**53.77**
+Overlap	**63.00**	**49.91**	**55.70**
+Chunk	**62.66**	**50.34**	**55.83**
+Dependency	**63.55**	**50.51**	**56.29**
+Parse Tree	**63.77**	**50.26**	**56.21**
+SQLF	**64.15**	**50.77**	**56.68**
+WordNet Synset	**67.11**	**51.79**	**58.47**

It can be seen from the table, the performance improves as more features are introduced, until the F1 measure reaches 58.47% which is comparable to the reported best results (55.5%) of [10] on the ACE2003 training data.

From the new features introduced in this work, the POS tag features and the general semantic features all contribute to the improvement. The improvement of the general semantic features, including semantic features from SQLF and WordNet Synset, is rather significant -- 2.26 % (from 56.21% to 58.47%), while the influence brought by some syntactic features such as the chunk, dependency and parse tree features is only 0.51% (from 55.70% to 56.21%) in total. Therefore, the contribution of the semantic

features shows that general semantic information is beneficial for relation extraction and should receive further attention.

We also tried replacing the synset features with the hypernym features, which led to 67.54% in Precision, 50.85% in Recall and 58.02% in F1. It shows that using the hypernym instead of the synset can increase precision but harms recall, which leads to a slight fall in F1. Consequently in the following experiments, we only consider the synset features.

The entity features lead to the best improvement in performance. Actually here we use only 3 features, among which entity subtype and class are newly introduced. Further investigation shows that the two new features are in fact very effective: among the overall F1 improvement (13.48%), the entity type feature contributes 9.29% and the two new features contribute further 4.19%. This result shows that the more accurate information of the entity mentions we have, the better performance can be achieved in relation extraction.

From Table 15, we can also see that the impact of the deep features is not as significant as the shallow ones. Zhou et al. [10] show that chunking features are very useful while the dependency tree and parse tree features do not contribute much. Our results even show that features from word, POS tag, entity, mention and overlap can achieve 55.70% F1, while the deeper features (including chunk, dependency tree, parse tree and SQLF) only give around 1% improvement over simpler processing. As the number of features impacts directly the required size of training data and training and application efficiency (more features need more annotated data for training the model and need more computation resources), there is an interesting trade-off in feature selection for relation extraction.

5.3 Experiments on Hierarchical Classification

As already discussed, ACE2004 defined both an entity and relation hierarchy, which provides a data resource for evaluating our method for ontology-based IE. The significant contribution of entity subtype and class features demonstrated above shows that the entity hierarchy information is important for relation extraction. As shown in Fig. 1 the relation hierarchy has three levels, so we ran experiments to evaluate our method with these different classification levels: *subtype classification* -- 23 relations at leaf level, *type classification* -- 7 relations at middle level, *relation detection* -- predicating if there is relation between two entity mentions, which can be treated as a binary classification task. The experiments on the three different classification levels have been done separately. In each experiment, the classifier is trained and tested on the corresponding relation labels (i.e., 23, 7, or 1 relations). Table 16 shows the overall results using the complete feature set listed in Table 5:

Table 16. The result on different classification levels.

Levels	Precision	Recall	F1
Subtype classification	**67.11**	**51.79**	**58.47**
Type classification	**67.07**	**61.11**	**63.95**
Relation detection	**70.28**	**69.80**	**70.04**

The results show that performance on relation detection level is the highest while that on subtype classification is the lowest. The Precision, Recall and F1 all show the same

trend, revealing that it is more difficult to classify on deeper levels of the hierarchy because there are less examples per class and also the classes are getting more similar as the classification level gets deeper. This has been supported by the more detailed results for the relations type *EMP-ORG* and its subtypes, as shown in Table 17. The performance for the type *EMP-ORG* when classifying on the type level is the best among all 7 relation types: 75.19% Precision, 72.62% Recall and 73.88% F1. However, the performance on the 7 subtypes within *EMP-ORG* when classifying at subtype level is not only much lower than the result for *EMP-ORG* overall but also rather unstable: from zero for *Partner* to 71.93% for *Subsidiary*. The two biggest subtypes *Employ-Exec and Employ-Staff* get only 67.23% F1 which is much lower than the 73.88% on type level for its parent type *EMP-ORG*. We consider that the zero result for *Partner* is mainly due to too few instances. Therefore, the closer distance between the classes at subtype level causes the performance to decrease and become unstable.

Table 17. The result on the subtypes of EMP-ORG.

Subtypes	Num. of instances	Precision	Recall	F1
Employ-Exec	**128**	**72.73**	**62.50**	**67.23**
Employ-Staff	**132**	**70.18**	**64.52**	**67.23**
Employ-Undetermined	**35**	**77.78**	**50.00**	**60.87**
Member-of-Group	**40**	**73.08**	**47.50**	**57.58**
Subsidiary	**62**	**78.85**	**66.13**	**71.93**
Partner	**3**	**0.00**	**0.00**	**0.00**
Other	**9**	**25.00**	**11.11**	**15.38**

Table 18. The F1 results on different feature sets and classification levels.

Features	Relation detection	Type classification	Subtype classification
Word	**59.27**	**41.26**	**34.85**
+POS Tag	**60.39**	**45.79**	**38.68**
+Entity	**64.25**	**57.88**	**52.16**
+Mention	**65.97**	**58.22**	**53.77**
+Overlap	**67.82**	**61.69**	**55.70**
+Chunk	**67.14**	**61.48**	**55.83**
+Dependency	**71.80**	**63.22**	**56.29**
+Parse Tree	**71.38**	**63.28**	**56.21**
+SQLF	**70.96**	**63.18**	**56.68**
+WordNet Synset	**70.04**	**63.95**	**58.47**

We also investigated the influence of different feature sets on the different classification levels (see Table 18). The best performance on relation detection resulted from the combination from word to dependency features and further features from parsing, SQLF and WordNet lead to performance decrease. However, such phenomenon does not appear at the type and subtype levels, in which the improvements are almost stable as more features are introduced and the best performance is achieved with the complete feature set. Such difference suggests that the deeper the classification level is, the more significant the effect of the features is and more syntactic and semantic features are needed. The difference in the improvement at the different levels also supports this hypothesis: as more features are used, the improvement in relation detection is only 11.53% (from 59.27% to 71.80%), while improvement in type and subtype classification is much more significant:

22.69% (from 41.26% to 63.95%) and 23.62% (from 34.85% to 58.47%). Furthermore, the impact of the SQLF and WordNet synset features shows that semantic knowledge will play more important role in extracting finer granularity relations.

5.4 Learning Curves

Learning curves are an important factor in evaluating ML methods, especially for supervised IE, because it requires annotated training data which is typically hard to obtain. As ontology-based IE is becoming a focus of intensive research, the lack of data annotated according to an ontology causes an even greater challenge.

We apply our method with different feature sets and various sizes of training instances to investigate the learning curves and features. The feature set has been divided into two parts: one is called *shallow features* consisting of word, POS tag, entity, mention and overlap features; the other is *deep features* including chunk, dependency, parsing tree and semantic ones. In the following experiments, we measure the classifier using shallow features, deep features and all features separately and the training instances increase from 1,000 to all. Figures 3, 4 and 5 show the learning curves of the F1 measure for classification on subtype, type and relation detection.

From those figures we can see that deep features on their own never lead to a better performance than the shallow ones which confirms the conjecture that shallow features are more useful. Nevertheless, when all features are used together, the performance improves above that of shallow features on their onw, when the number of training instances grows. Although there are few exceptions, Figures 3, 4, and 5 show clearly that the deeper the classification level is, the more training instances are needed for all features to outperform the shallow features. When classifying at subtype level (41 classes), the performance of shallow features increases faster until the training instances get to 20,000, after which the total feature set outperforms the shallow features (Fig 3). When classification is carried out only at type level (i.e., 7 classes) after only 2,000 examples the total features outperform the shallow ones (Fig 4, there is an exception at 5,000, after which the trend is stable). For the simplest task, i.e., relation detection, even 1,000 training instances are enough for the classifier to work better with all features than only the shallow ones (Fig 5, there is also only one exception at 10,000).

These results suggest that with the consideration of the data sparseness problem, as the classification level gets deeper, more features require more training data. If the classification level is shallower, we need less training data to make all features work better. However, if the classification level is very deep, shallow features can work better separately than together with the deep features, especially when there is not enough training data. Therefore, given that large scale ontology-based relation extraction usually suffers from lack of training data, it seems more beneficial to use shallow features at least initially, until the number of training examples is such that the classifier using all features outperforms the one using only shallow ones.

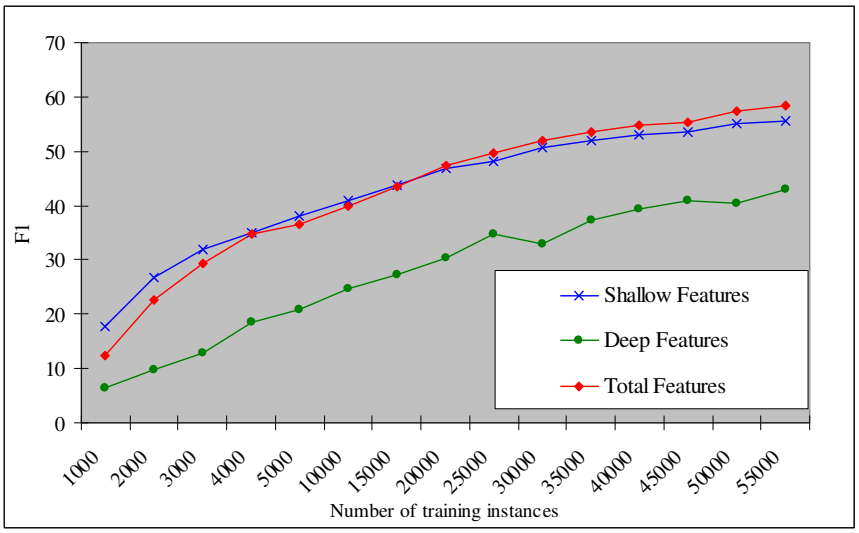

Fig. 3. The learning curves for subtype classification.

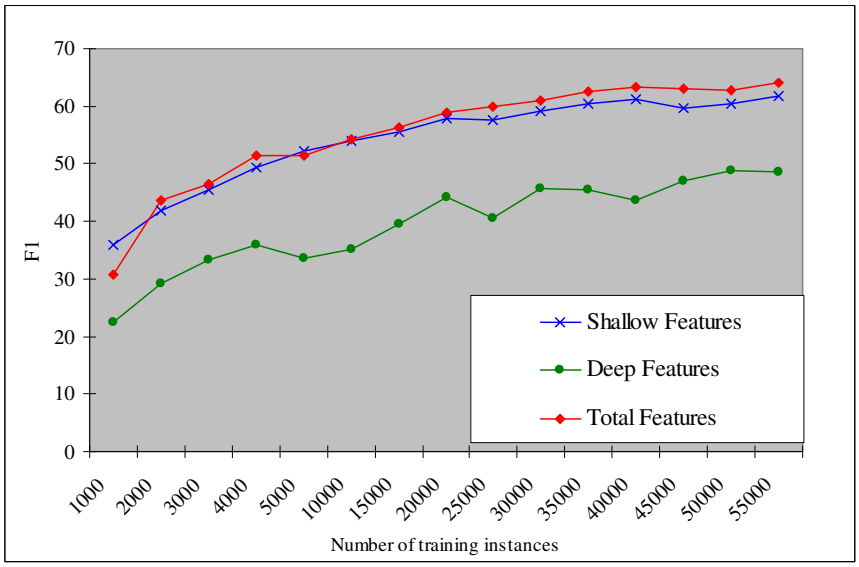

Fig. 4. The learning curves for type classification.

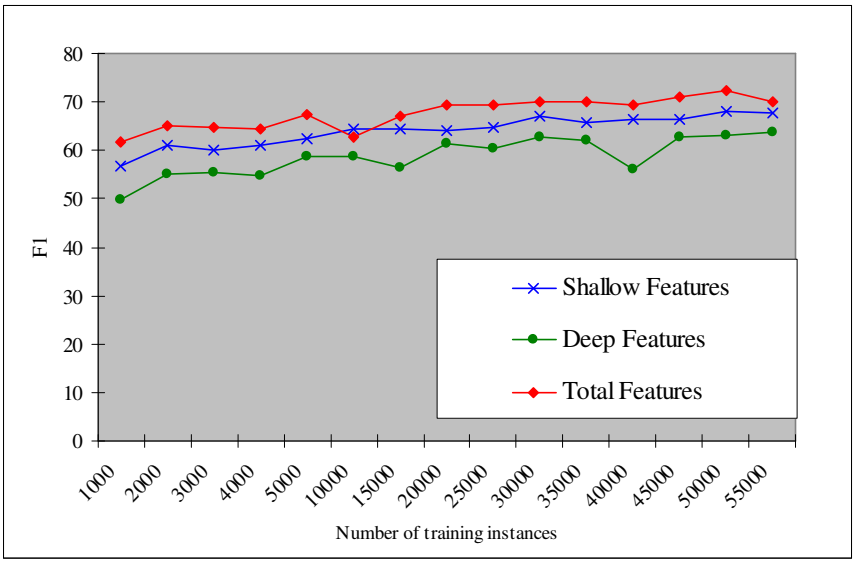

Fig. 5. The learning curves for relation detection.

6 Conclusions

In this deliverable, we used SVM-based classification for relation extraction, using features produced by NLP tools provided within GATE. In comparison to previous work, we introduce some new features, including POS tag, entity subtype and class features, entity mention role features and even general semantic features which all contribute to performance improvements. Experiments shows, for relation extraction work and as the classification level gets deeper, the features containing more information such as semantic ones can give more significant contribution. We also investigated in detail the impact of various factors on performance: the features, the classification levels and the amount of training data, which shows there is a trade-off among these factors for relation extraction work.

Further research needs to be carried out in two directions. Firstly, as the number of features used for relation extraction is very large, we intend to investigate which features affect most the performance. To this end, we will apply some ML technologies for feature selection, which will help us identify more discriminating features. Secondly, though the ACE2004 entity and relation type system provides a hierarchy organization which is somewhat like an ontology, it is still very limited for large-scale ontology-based IE, due to the small number of relations covered.

Bibliography and references

1. Appelt, D.: An Introduction to Information Extraction. Artificial Intelligence Communications, 12(3) (1999) 161–172
2. Handschuh, S., Staab, S., Ciravegna, F.: S-CREAM --- Semi-automatic CREAtion of Metadata. Proceedings of the13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Siguenza, Spain (2002)
3. Motta, E., Vargas-Vera, M., Domingue, J., Lanzoni, M., Stutt, A., Ciravegna, F.: MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Siguenza, Spain (2002)
4. Freitag, D., and McCallum A.: Information extraction with HMM structures learned by stochastic optimization. Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00), 584–589,Menlo Park, CA. AAAI Press (2000)
5. Lafferty, J., McCallum, A., Pereira. F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proc. 18th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA (2001) 282–289
6. Kambhatla, N.: Combining lexical, syntactic and semantic features with Maximum Entropy models for extracting relations. Proceedings of 42th Annual Meeting of the Association for Computational Linguistic. 21-26 July Barcelona, Spain (2004)
7. Vapnik, V.: Statistical Learning Theory. John Wiley (1998)
8. Zelenko, D., Aone, C., Richardella, A.: Kernel methods for relation extraction. Journal of Machine Learning Research (2003)1083-1106
9. Culotta, A., Sorensen, J.: Dependency tree kernels for relation extraction. Proceedings of 42th Annual Meeting of the Association for Computational Linguistics. 21-26 July Barcelona, Spain (2004)
10. Zhou G., Su, J., Zhang, J., Zhang, M.: Combining Various Knowledge in Relation Extraction, Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (2005)
11. ACE. http://www.nist.gov/speech/tests/ace/
12. Annotation Guidelines for Entity Detection and Tracking (EDT) Version 4.2.6, http://www.ldc.upenn.edu/Projects/ACE/docs/EnglishEDTV4-2-6.PDF. (2004)
13. Annotation Guidelines for Relation Detection and Characterization (RDC) Version 4.3.2, http://www.ldc.upenn.edu/Projects/ACE/docs/EnglishRDCV4-3-2.PDF. (2004)
14. Hsu, C.-W., Lin, C.-J.: A comparison of methods for multi-class support vector machines , IEEE Transactions on Neural Networks, 13(2). (2002) 415-425
15. Miller, A., “WordNet: An On-line Lexical Resource”, Special issue of the Journal of Lexicography, vol. 3, no. 4 (1990)
16. Gaizauskas, R., Hepple, M., Saggion, H., Greenwood, M.A., Humphreys, K.: SUPPLE: A Practical Parser for Natural Language Engineering Applications. Technical report CS--05--08, Department of Computer Science, University of

Sheffield (2005)

17.Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics. Philadelphia, July (2002)

18.Lin, D.: Dependency-based Evaluation of MINIPAR. In Workshop on the Evaluation of Parsing Systems, Granada, Spain, May (1998)

19. McCarthy, D., Koeling, R., Weeds, J. and Carroll, J. (2004) Finding predominant senses in untagged text. In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics. Barcelona, Spain. pp 280-287.

20. N. Chinchor. Muc-4 evaluation metrics. In Proceedings of the Fourth Message Understanding Conference, pages 22-29, 1992.

Appendix I
Guide to Running Relation Extraction on ACE 2004 Training Data

This guide is organized as follows:

- Section 1) describes how to install the deliverable and gives an overview of its directories.
- Section 2) introduces the simple command which runs the experiments for relation extraction task on the ACE 2004 training data.
- Section 3) provides more information on the `data` directory which contains the original and the processed versions of the ACE 2004 training data.
- Section 4) and section 5) give information on how to produce the ACE 2004 training data.
- Section 6) and section 7) provide information on how to prepare the training and the testing data files and how to use LIBSVM to train the SVM model and to apply it to classify the testing data.

1) How to Install

You should download the deliverable and unzip it into some directory. It has six sub directories:

- `app`: It contains the GATE JAPE and the application files for converting the ACE 2004 training data from its original version to the GATE documents that contain all the ACE annotations plus all the NLP annotations.
- `bin`: It contains the JAR for the relation extraction task, batch script for running experiment in command mode under the Windows.
- `data`: This directory is created as a placeholder for the ACE2004 training data and its processed versions for the relation extraction task.
- `doc`: It contains the deliverable document.
- `lib`: It contains other required libraries. Make sure they are on your CLASSPATH
- `src`: It contains source code for the relation extraction task.

Note: Since the ACE2004 training data is distributed as LDC2005T09 ACE 2004 Multilingual Training Corpus, you need to obtain a license for it from LDC (http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T09).

2) How to Run

Before running the experiment, you should make sure that:

- GATE has been installed and gate.jar is on the CLASSPATH.
- WordNet 2.0 has been installed. file_properties.xml (under `bin`) should be modified to include the location of your local copy of the WordNet 2.0 index files, for example:

  <param name="dictionary_path" *value*="c:\program files\wordnet\2.0\dict"/>
- All libraries under the `lib` directory and the re.jar from the `bin` directory appear on the CLASSPATH.

Then change to the `bin` directory, execute the batch file `run.bat` to run the training and testing procedure and to get the result.

Usage:
`run` <data-url> <classification-level> <feature-set-file> <number-of-training-examples> <report-file-name>

- data-url: URL of the data directory that contains the ACE2004 training data and its processed versions for the relation extraction task
- classification-level: classification level
  0 -- relation level, i.e. relation detection
  1 -- relation type level, i.e. classify relation type
  2 -- relation subtype level, i.e. classify relation subtype
- feature-set-file: features to be used in the SVM classifier
- number-of-training-examples: number of examples used for training, maximum at 54457
- report-file-name: file to be used for saving the classification results

e.g.
```
> run file://D:/RE/data 2 features.txt 5000 result.txt > log.txt
```

3) `data` Directory

The `data` directory should contain the original ACE2004 training data and the data processed by GATE for the relation extraction task:

- `BNEWS`: broadcast news from the TDT-4 Corpus
- `NWIRE`: newswire data from the TDT-4 Corpus
- `ARABIC_TREEBANK`: Arabic Treebank 1 Corpus - English translations from the MT-2003 translation data set
- `CHINESE_TREEBANK`: Chinese Treebank Version 4 - English translations from the Chinese Treebank English Parallel Text Corpus
- `FISHER_TRANSCRIPTS`: Fisher Telephone Speech collection

Each of the above directories contains five subdirectories:

- `SGM`: source text files from the ACE 2004 training data.
- `APF`: AFP files from the ACE 2004 training data, providing the ACE annotations.
- `NewSGM`: the modified SGM files
- `clean`: all files under the `NewSGM` should be saved as GATE documents (using save to XML()) under this folder.
- `marked-all`: the GATE XML documents containing all annotations from the APF files related to entity and relation.
- `Parsed-all/DataStore`: All files under the `marked-all` directory should be saved under a serial datastore under this directory. These files need to be processed with various NLP tools (using a GATE application).

4) How to Convert Original ACE2004 Training Data to GATE Documents

After the ACE2004 data files (from `ACE2004\data\English directory`) have been copied to some local directory (e.g. `ACE2004`), user should run the following batch file:

Usage:
`bin\init-data.bat` <the absolute path of the `ACE2004 directory`>

It performs the following actions:

- creates all required directories under the `data directory`;
- copies the source (ACE2004 training data) SGM text files into the corresponding `SGM` directory and the source APF files into the corresponding `APF` directory;
- modifies the files in each `SGM` directories to replace all strings ***&*** with string ***;AMP;*** and ***&*** with ***&*** . All the processed files (including the unchanged files) are saved into the `NewSGM` directory.

In order to convert the original ACE2004 Training Data into GATE Documents, the following actions should be taken for each of the five subdirectories (i.e. `bnews, nwire, arabic_treebank, chinese_treebank and fisher_transcripts`):

- Populate corpus with files with extension ".sgm" from the directory `NewSGM` in GATE, and save them as XML into the `clean` directory.
- Process each document with the `apf-entity-all-to-gate.bjape` and the `apf-relation-to-gate.bjape (binary jape files)` (distributed under the `app` directory):
    - Populate a corpus (e.g. *cleanfiles*) with ".xml" files from the `clean` directory.
    - Populate a corpus (e.g. *apffiles*) with ".xml" files from the `APF` directory.
    - Using the `apf-entity-all-to-gate.bjape` and the `apf-relation-to-gate.bjape` (binary jape files) and setting their inputASName parameter to the "*Original markups*", process the *apffiles* corpus.
    - Save the corpus *cleanfiles* in the GATE XML format under the `marked-all` directory.

5) How to Process the GATE Documents with NLP Tools

Make sure you have Minipar and Buchart installed on your system. (*Note: Refer to the gate userguide at http://www.gate.ac.uk/sale/tao/index.html for the information on how to compile these resources.*) For each of the five corpora (in `bnews, nwire, arabic_treebank, chinese_treebank and fisher_transcripts`),

- load the documents from `marked-all` directory into a GATE corpora (e.g. *marked)*
- copy the GATE application files `NLPApp.gapp` and `NLPApp-single-nl.gapp` (distributed under the `app` directory) into the GATE root directory. For the data under `fisher_transcripts` use the `NLPApp-single-nl.gapp`.
- restore the application file `NLPApp.gapp` (and `NLPApp-single-nl.gapp for the fisher_transcripts`) in GATE.
- Process the `marked` corpus.
- Create a SerialDatastore under the `Parsed-all/DataStore` directory and save the corpus in it.

The serial data stores are used to generate the training and the testing examples.

6) How to Convert GATE Documents (ACE2004 Training Data) into the Training and the Testing Data Files for LIBSVM Classifier

`bin/run.bat` should be executed in order to convert GATE documents into the Training and the Testing data files. It produces the training and testing data files for LIBSVM classifier. `bin/run.bat` performs the following actions:

- Generates nominal vector files from the GATE documents

Batch file `prepare-nominal-vectors.bat` generates nominal vectors for each GATE XML document available under the `data/XXX/Parsed-all/DataStore` (where XXX is one of the `ARABIC_TREEBANK`, `BNEWS`, `CHINESE_TREEBANK`, `FISHER_TRANSCRIPTS` and `NWIRE`). It then splits these vectors into two files `total-train.vct` and `total-test.vct` which represent the training and the testing set for classification respectively. The usage of the batch file is:

Usage:
`prepare-nominal-vectors.bat` <data-url> <classification-level>
- data-url: the url of the data directory which contains the ACE2004 training data and its versions processed for the relation extraction task
- classification-level: classification level
  0 -- on relation level, i.e. relation detection
  1 -- on relation type level, i.e. classify the relation type
  2 -- on relation subtype level, i.e. classify the relation subtype

e.g.:
```
>prepare-nominal-vectors.bat  file://D:/RE/data  2
```

- Select training examples

You can select how many examples from the total training data should be used for training.

Usage:

`select-training-data.bat` <total-training-file> <number-of-training-examples-to-select> <result-file>

- total-training-file: specifies which file to select the training examples from
- number-of-training-examples-to-select: the number of examples used for training, maximum at 54457
- result-file: specifies where to save the selected training examples

e.g.

```
>select-training-data.bat total-train.vct 30000 train.vct
```

- Select features

You can select what features to use for classifying for both the training and the testing data.

Usage:

`select-features.bat` <feature-set-file> <training-source-file> <training-dest-file> <testing-source-file> <testing-dest-file>

- feature-set-file: specifies the features to be used
- training-source-file: specifies the file to select the training examples from
- training-dest-file: specifies the destination file to save the training examples that contain features specified in the feature-set-file
- testing-source-file: specifies the file to select the testing examples from
- testing-dest-file: specifies the destination file to save the testing examples that contain features specified in the feature-set-file

e.g.

```
>select-features.bat feature-set.txt train.vct train_f.vct total-test.vct test_f.vct
```

- Nominal to Numeric examples

As LIBSVM can only deal with numeric attributes, the nominal examples (for both training and testing data) need to be converted into the numeric examples.

Batch file `nominal-to-numeric.bat` should be used to convert the nominal examples into the numeric ones:

Usage:

`nominal-to-numeric.bat` <training-nominal-file> <training-nominal-file> <testing-nominal-file> <testing-numeric-file>

- training-nominal-file: the file containing the nominal training examples
- training-numeric-file: the file to save the converted numeric training examples
- testing-nominal-file: the file containing the nominal testing examples
- testing-numeric-file: the file to save the converted numeric testing examples

Note: this command produces a configuration file (named map.cfg) which contains the mapping information.

e.g.

```
> nominal-to-numeric.bat  train_f.vct train_f.bvt test_f.vct test_f.bvt
```

7) How to Use LIBSVM to Classify Examples

- Training

Usage:

`svm-train` -t 0 <training-numeric-file> <svm-model-file>

- training-numeric-file: name of the file that contains numeric training examples
- svm-model-file: name of the SVM model file
- The option -t 0 is required in order to use the linear kernel.

e.g.,

```
>svm-train -t 0 training.bvt  ace.mdl
```

There are two files, ace-subtype-all-features.mdl (a model) and ace-subtype-all-features.cfg (a configuration file), which are provided under the bin directory. The model is already trained on the total training set (total-train.vct) considering features from the features.txt. One can use these files to classify relations at a sub-type level.

- Testing

Usage:

`svm-predict` <testing-numeric-file> <svm-model-file> <output-file>

- testing-numeric-file: name of the file that contains the numeric testing examples to be classified
- svm-model-file: name of the SVM model file
- output-file: name of the file used to store classification results

e.g.,

```
> svm-predict testing.bvt ace.mdl output.txt
```

- Evaluation

Usage:

`stat` <testing-numeric-file> <output-file map-config-file> <report-file>

- testing-numeric-file: name of the file that contains the numeric test example
- map-config-file: mapping configuration file
- resport-file: the text file to save the report

e.g.

```
> stat testing.bvt output.txt  map.cfg result.txt
```