

# [The natural language processing tools english language essay](https://assignbuster.com/the-natural-language-processing-tools-english-language-essay/)

[](https://assignbuster.com/)[Linguistics](https://assignbuster.com/essay-subjects/linguistics/), [English](https://assignbuster.com/essay-subjects/linguistics/english/)

resolving coreferential mentions in blog commentsTable of ContentsList of FiguresIntroduction

## Background

Thecoreference is defined in linguistic as the grammatical relation between two words that have a common referent. In computational linguistics, coreference resolution is related to discourse. Pronouns and other referring expressions should be placed together into an equivalence class in order to correctly interpret the text or to estimate the importance of various subjects[1]. Most blogs allow users to post comment after each article, and so do news websites with dynamic content. User posts are often very short and typically have slang, spelling errors, and creative use of language. We are investigating how user comments relate to the news/blog articles, in particular focussing on developing an algorithm for automatically linking words in the comments with words in the original article when they refer to the same person, place or thing. At times it’s easy to identify coreferential text, for example, when a person’s name is given in full, but is made harder due to different forms of address (e. g., Gorden Brown, the MP for Kirkcaldy, the ex-prime minister, Brown, etc) or use of anaphora (he, she, they, it, etc). Accurately predicting of the referent is important when searching over the data, or for later processing to determine the meaning or sentiment of user comments/Tweets. To realise our objective we started by creating a data set for studying and evaluating the phenomenon, and then developing algorithms for predicting the correct referent. The algorithms make use of machine learning classification algorithms such as logistic regression, support vectors machines, and conditional random fields in order to learn a predictive model of coreference from data. The processing of un-annotated text is the ultimate goal for a coreference system.

## Problems

## ........

## Research Strategy and Objectives

The research objectives of this thesis are: Conduct an investigation usingStanford coreferencing tool to highlight theweaknesses and shortcomings in the coreference resolution of mentions in blog comments, if any. This is carried out with three purposes -to develop an algorithm for automatically linking words in the comments with words in the original article when they refer to the same person, place or thing; to establish a sub-standard benchmark results for about 12 articles with user commentsby doing the coreferencing manually as well as to gauge the performance of the Stanford tool. Provide a solution for the weaknesses and shortcomings found in the investigation. This is accomplished....................

## Report Overview

There are in total of 6 chapters. Chapter 1 covers background information, the problems, and research strategy and objectives. conducts literature review on coreferencing tools. An investigation usingStanford coreferencing tools is conducted in with the use of the information published in literature. presents the main contributions of this thesis; a new ............ is proposed and its effectiveness demonstrated and compared against ........... the above mentioned coreferencing tools. is where conclusions are drawn. Potential future work is suggested in . Literature Review

## Coreference resolution

In the community of natural language processing the importance of identifying all mentions of entities and events in text and clustering them into equivalence classes is well recognized. The methods of achieving this task evolved in the last two decades. The first notable results on resolution of coreference, using corpus-based methods, belonged to McCarthy and Lenhert (1995)[2]who have experimented applying hand-written rules to decision trees. After the systematic study on the decision trees conducted by Soon and all [3]improvements in the field of language processing and learning techniques followed. New knowledge sources, such as shallow semantics, static ontology and collaboratively built encyclopaedic knowledge resources, were recently exploited (Ponzetto and Strube, 2005, [4]; Ponzetto and Strube, 2006, [5]; Versley, 2007, [6]; Ng, 2007, [7]). The current techniques rely primarily on surface level feature, syntactic features, and shallow semantic features. Currently new models and algorithmic techniques are being developed for the coreference resolution, but only few of them rely on strong features to sustain the pairwise baseline. The de facto standard datasets for current coreference studies are the Message Understanding Conferences (MUC) (Hirschman andChinchor, 1997,[8] ; Chinchor, 2001,[9]; ChinchorandSundheim, 2003,[10])and the ACE (G, Doddington et al, 2000, [11]) corpora. Both MUC, whichwas tagged with coreferring entities identified by noun phrases (NPs) in the text and which represents small training and test sets, and ACE, which has much more annotation, but is restricted to a limited subset of entities, are less consistent, in terms of inter-annotator agreement (ITA) (Hirschman et all, 1998, [12]), and thus diminish the reliability of the predictive models given by the classifier deriving the data on the statistical evidence in the form of lexical coverage and semantic relatedness. WordNet, an external resource, can add a layer that helps the system to recognize semantic connections between the mentions in the text. The task of entities and events coreference constitutes an ongoing research. Standard parameters and testing sets for the evaluation, based on new resources providing multiple integrated annotation layers (parses, semantic roles, word senses, named entities and coreference), and that support joint models, have become a strong trend. Researchers have lately concentrated on the new models and algorithmic techniquesdevelopment for solving the coreference resolution problem. Messages (such as consumer reviews, blogs, e-mails, short messages, etc.) are wide spreading on the Internet, and thus, the developing of technologies to extract opinions, evaluations, beliefs and speculations from text, becomes crucial. Newspapers publish their news articles online and offer their readers the opportunity to publish their own comments and opinions about an article. News became interactive by updating news stories every time an event evolves and by offering live blogs to people, for direct communication with the journalists present at the scene of the event. People comments are useful for making everyday decisions, such as which brand to choose, which movie to go to, which hotel to choose, etc. The information potentially interesting in fields such as product and service benchmarking, marketing researches and advertising, customer complaint management, or customer relation management, requires an automatic media reviewing procedure, as handling the information manually by media analysts is impossible. Newspaper articles and reader comments are related through the topic. Blogs can be seen as low structured and unedited online diaries, while the published news articles are highly structured, factual and edited. Blog posts, shown in chronological order, and having the most recent listed first, contain usually links to other pages and spelling errors, ungrammatical sentences, abbreviations and punctuation marks denoting feelings. Comments on articles and blogs are sharing the same informal writing style and structural characteristics. An important feature of blogs is the timeline. Aiming opinion summarization, Stoyanov and Cardie[13]focused on identifying opinion holders and resolving coreference relations between them, by using partially annotated data only for the opinion holder's coreferential information. The opinion mining systems werefurther improved by enhancing the entities recognition and by using information on coreference relations between opinion holders and entitiesrepresenting opinions or beliefs. Features designed to handle spoken dialogue data were also beneficial for recall in coreference resolution on blogs and commented news. Opinion mining and anaphora resolution are similar type of tasks if we consider linking an opinion to its source as similar with linking an anaphor to its antecedent. Systems for handling anaphora in multi-person dialogues, integrating different constraints and heuristics, can give also useful insights for coreference resolution on blogs. In analyzing and comparing the available reports on coreference resolution, the relevance and significance of the content covering the coreference resolution, which is denoted by documents’ length, was used as a first criterion.

## Coreference task issues

In defining a coreference task are involved many different parameters. The evaluation criteria and the training data used evolved in time, making difficult for researchers to clearly determine the coreference’s state of the art or which particular areas require further attention. Datasets available for each task were limited in size and scope. Mentions are often heavily nested, making difficult their detection. The results of evaluating a system against a gold standard corpus might be affected by mismatches in mention boundaries, or missing mentions. The metonymy’s phenomenon raised a problem for coreference relations, as the relation was annotate and recognize before or after coercion. Comparative results, that differ in the entities’ types and coreference annotated, were published for OntoNotes[14] corpus, ACE [15], and MUC[16]corpora. ACE corpus evolved in time, its task definition was changed and the studied cross-sections were different from a research to the next one making it hard to clear up and interpret the results. The choice of coreference evaluation metrics remains a tricky issue as each of them tries to address the shortcomings or biases of the earlier metrics. In OntoNotes coreference task the spoken genres were treated with perfect speech recognition accuracy and perfect speaker turn information, which are not realistic application conditions.

## Coreference task in OntoNotes

OntoNotes[14] is providing a corpus of general anaphoric coreference having an unrestricted set of entity types and additional layers of integrated annotation that is capturing additional shallow semantic structure. A rich integrated annotation allows a better automatic semantic analysis for cross-layer models, but demands a strong storing mechanism while providing efficient access to the underlying structure. OntoNotes uses a relational database representation capturing inter- and intra-layer dependencies and providing an object-oriented Application Platform Interface (API), ensuring an efficient access to data. Integrated predictive models having cross-layer features can make use ofOntoNotes annotations (as approximate " 1. 3M words has been annotated with all the layers"[14]). OntoNotes is a multi-lingual resource/corpus having multiple layers of annotation covering three languages: English, Chinese and Arabic, but the " CoNLL-2011 shared task was based on the English portion of the OntoNotes 4. 0 data." [14]In the OntoNotescoreference is distinguishedthe Identical (IDENT) type, for anaphoric coreference, that linksproper pre-modifiers, dates and monetary amounts, pronominal, named, or definite nominal mentions of specific referents, while excludes mentions of generic, underspecified, or abstract entities, proper nouns that are in a morphologically adjectival form, which are treated as adjectives. Verbs coreferenced with a NP, including morphologically related definite nominalizations and definite NPs that refer to the same event, or with another verb, are added as single-word spans, for convenience. All pronouns and demonstratives, excepting the generic you, but including those in quotedspeech, are marked. Expletive or pleonastic pronouns (it, there) are not considered for tagging, and are not marked. Generic nominal mentions are not liked between them. Bare plurals are considered generic. Two generic instances of the same plural noun from successive NPs are marked as distinct IDENT chains. Deictic and other temporal expressions related to the time of the article/textwriting are coreferenced by using knowledge from outside the text. Dates embedded in multi-date temporal expressions are not separately connected to other date’s mentions; andtheAppositive (APPOS) type, which function as attributions, links a head, or referent (a NP pointing to an object or concept, modifying the immediately-adjacent noun phrase, renaming or further defining the first mention) with attributes of that referent. The order of head marking is scaled from proper noun, pronoun, definite NP, indefinite specific NP to non-specific NP, starting from the left-most member of the appositive, and including the definite marker (the) or possessive adjective (his) companion. Nested NP spans are not linked. The process of annotation starts by extracting automatically mentions of the NPs from the Penn Treebank. The relationship between attributes signalled by copular structures and their referent are captured through word sense. Subject complements that follow copular verbs such as: be, appear, feel, look, seem, remain, stay, become, end up, get, etc. and small clause constructions are not marked as IDENT or APPOS coreference. Geo-Political Entity (GPEs)is always coreferenced. Organizations are not linked with its members. The coreference task was to automatically identify entities and events mentions in text and to link the coreferring mentions together to form entity/event chains, using automatically predicted information on the other structural layers. There were two tracks: the closed track; where systems used only the provided data, and a pre-computed number and gender table by Bergsma and Lin (2006)[17], to allow the algorithmic comparisons. WordNet use was allowed. Predicted versions and manual gold-standard of all the annotation’s layers were provided for the training and test data, and each system chose the best fitted of these two for the task. the open track, where systems used the provided data, and the same pre-computed number and gender table, WordNet and external resources such as Wikipedia, gazetteers etc. to get an idea of the best achieved performance on the task, even without getting a comparison across all systems. In the open task participated also research systems that depend on external resources. For the task, it was used a train/development/test partition which included the WSJ portion of the newswire data and other partitions. The newswire in OntoNotes contains WSJ data and Xinhua news. The list of training, development and test document IDs were available on the task webpage.[18]. The documents were split in smaller parts that were as separate documents, in order to be efficiently annotated. The majority of sub-token annotation and the traces from the syntactic trees disappear with the Penn Treebank revision. The remaining sub-token annotationwas ignored for the task. The annotation was revised to include propositions for " be" verbs. The disfluencies were also removed from the OntoNotes parses. PropBank was also revised, and enhanced by the addition of LINKs that represent pragmatic coreference (LINK- PCR) and selectional preferences (LINK-SLC) as a part of the OntoNotes DB Tool. On the task page for the participants was provided " a revised version of the sense inventories, containing mapping to WordNet 3. 0"[14]. 18 Name types were specified as Named Entities. Discourseinformation for correctly linking anaphoric pronouns with their right antecedentswas provided in the form of a column in the ". conll" table, assuming there is only one speaker/writer per sentence. For the predicted annotation layers were used trained automatic models producedusing the retrained Charniak parser (Charniak and Johnson, 2005, [19]). A tested word sense tagger, having performances not comparable with previous literature, was available for the test set. A modified ASSERT(Pradhan et al., 2005, [20]), with two-stage mode for filtering out the NULL arguments and to classify NON- NULL arguments using ten classifiers, was used to predict propositional structure. To generate the scores was used the CoNLL-2005 scorer. To predict the named entities was used BBN’s IdentiFinderTM system which used a pre-trained model having a catalogue of name types which omit the OntoNotes NORP type (for nationalities, organizations, religions, and political parties). For the task a database representation was created, along with a Python API (Pradhan et al., 2007a, [20])" In the OntoNotes distribution the data isorganized as one file per layer, per document" [14]. To remove the EDITED phrasessome of the trees for the conversation data were dissected. The noun phrase which satisfies the markable definition in an individual corpus is a mention (or a markable). The pair of coreferential mentions is related by a link. A mention without links to other mentions is called a singleton. The criteria used for the evaluation were: the counting of the correct answers for propositions, word sense and named entitiesthe use of several established metrics, which weight different features of a proposed coreference pattern differently, for parsingthe exact spans of mentions for determining correctness of the mentions granularityThetests input conditions were: predicted only (official)predicted plus goldmention boundaries (optional, as boundaries alone provide only very partial information), andpredicted plus gold mentions (supplementary, to quantify the mention detection impact on the overall task and the results when the mention detection is perfect)The mention detection score was considered only together with the coreference. The scores were established by using MELA metric, with CEAFe instead of CEAFm. ACE value was not considered for comparison. BLANC and CEAFmdid not factor into the official ranking score. The scorer, by design, removed singletons after the accuracy of mention detection was computed. Exact matches were the only ones considered correct. 18 systems submitted results for closed track and 5 systems for open track in the official tests. 9 systems submitted results for closed track and 1 system for open track in the optional tests, which revealed a bug in the automatic scoring routine, that could double-count duplicate correct mentions in a given entity chain by reporting the two mentions that identify the exact same token in the text as separate mentions. After the scorer fixing and re-evaluating all of the systems, only one system’s score was affected significantly. Gold mentions helped the systems to generate better entities, but the improvement in coreference performance was almost negligible. In the OntoNotes were used the mentions’ head words from the gold standard syntax tree to approximate the minimum spans that a mention must contain to be considered correct. The system’s performance was not much improved by using the relaxed, head word based, scoring. To solve the task most of the systems identified first the potential mentions in thetext using rule-basedapproaches (while only two used trained models), and then linked them to form coreference chains. One system used joint mention detection and coreference resolution. For predicting coreference were used various types of trained models, but the best-performing ones used the completely rule-based approach. NPs and pronouns are roughly 91% of mentions in the data, justifying that participants appear not to have focused much on eventivecoreference.

## Coreference task in ACE 2005

ACE [15] is a technology/programused for the automatic content extraction from source language data (in the form of natural text, and as text derived from ASR and OCR). ACE coreference task covers primarily the task of entities, values, temporal expressions, relations, and event’s recognition, and secondly it supports the entities, relation and events mention. ACE types of entities are: Person, Organization, Location, Facility, Weapon, Vehicle and Geo-Political Entity, each one having appropriate subtypes. In ACE the instance of a reference to an object is a mention. The collection of mentions referring to the same object in a documentis an entity. The required form of the output is " defined by an XML format call " APF" [15] available on NIST ACE web site[21]. The evaluation of ACE system’s performance was made for all the five primary tasks in all three languages, and included several types of sources (Newswire, Broadcast News, Broadcast Conversations, Weblogs, Usenet Newsgroups/Discussion Forum and Conversational Telephone Speech) and one processing mode (Document-Level, Cross-Document, Database reconciled). Performance on each ACE task was separately measured and scored using a model of the application value of system output. The overall value was determined as the sum of the each system output entity value which was computed by comparing its attributes and associated information with those of the reference that corresponds to it. The value was lost when system output information differed from that of the reference. Negative value resulted typically when system output was spurious. When the system output matched the reference without error, then it was achieved the perfect system output performance relative to which it was computed, as the system output information, the overall score of the system. According to the evaluation results, " the loss of value was attributable mostly to misses (where a reference has no corresponding system output) and false alarms (where a system output has no corresponding reference)"[15] or " due to errors in determining attributes and other associated information in those cases where the system output actually does have a corresponding reference" [15]. The ACE system’s performance on relations and events was affected by the system’s underlying performance on the arguments of relations and events, which include ACE entities, values and time expressions. Value and timex 2 elements were annotated only at the mention level, but their representation and evaluation was done considering that value and timex2 elements are globally unique and may have multiple mentions in multiple documents, therefore VAL’s and TERN’s evaluation and scoring was similar to that for entities. In the ACE Entity Detection and Recognition (EDR), the attributes used to refer to the entity are limited to the name(s) and only one entity type, one entity subtype, and one entity class, all of them being described in the annotation guidelines. Even if different entities may be referred to by the same name, such entities are regarded as separate and distinct. Their " determinations should represent the system’s best judgment of the source’s intention" [15] Each entity mention includes in its output its type, its head and its extent location, and optionally its role and style ( either literal or metonymic). EDR’s evaluation was designed to detect ACE-defined entities from mentions of them in the source language and to recognize and output the selected entity attributes and all associated information to these entities. All of the mentions of an entity were required to be correctly associated with that entity. The system output entity’s value was defined as " the product of two factors that represent how accurately the entity’s attributes are recognized and how accurately the entity’s mentions are detected." [15]

## ()

The EDR value score for a system was defined as being the sum of the values of all of the system’s output entity tokens divided by the sum of the values of all reference entity tokens, and thus, 100 percent was the maximum possible EDR value score. The ACE Value Detection and Recognition task (VAL) is limited to the values that are mentioned in the source language data, and only selected information are recognized. VAL is available only for Chinese and English languages. VAL’s evaluation was designed to detect ACE-defined value elements from mentions of them in the source language and to recognize and output the selected value attributes and all associated information to these entities. The ACE Time Expression Recognition and Normalization task (TERN) is limited to the temporal expressions that are mentioned in the source language data, and include the recognition of absolute and relative expressions, durations and event-anchored expressions, and sets of times. TERN’s evaluation was designed to detect ACE-defined timex2 elements from mentions of them in the source language and to recognize and output the selected timex2 attributes and all associated information to these entities. The ACE Relation Detection and Recognition task (RDR) is limited to the specified types of relations that are mentioned in the source language data . and only selected information on the relation between the two ACE entities called the relation arguments are recognized. When the ordering of the two entities does not matter, the relation between them is symmetric. The order matter for asymmetric relations and for the entity arguments must be assigned the correct argument role. The relation output, required for each document, includes information about its attributes (type, subtype, modality and tense), arguments (identified by a unique ID and a role), and mentions (sentence or phrase that expresses the relation). Good argument recognition ensures a good RDR’s and VDR’s performance. The Value of RDR system output entity was determined as the product of two factors representing the accuracy of the entity attributes’ recognition and the accuracy of the entity’s mentions detection. The system output relation value was defined as " the product of two factors that represent how accurately the relation’s attributes are recognized and how accurately the relation’s arguments are detected and recognized." [15]

## ()

The ACE Event Detection and Recognition task (VDR) is limited to the events that are mentioned in the source language data and only selected information are recognized. VDR is available only for Chinese and English languages. An ACE event involves zero or more ACE entities, values and time expressions. The event output, required for each document, includes information about its attributes (type, subtype, modality, polarity, genericity and tense), arguments (identified by a unique ID and a role), and mentions (the whole sentence or phrase that expresses the relation). The recognition of event mentions is not evaluated, but constitutes an allowed way for the system output events to map to reference events. The system output event value was defined as " the product of two factors that represent how accurately the event’s attributes are recognized and how accurately the event’s arguments are detected and recognized." [15]

## ()

The entity mention detection (EMD) uses formula identical to that for EDR, and each entity mention became an entity with only one mention. In relation and event mention detection (RMD and VMD) each relation and event mention becomes a separate and independent relation and event, which are evaluated in RDR and VDR. Mapping and scoring for RMD and RDR and also for VMD and VDR are different, as system output argument mentions became independent argument elements, while reference argument mentions remain unchanged as mentions of larger elements, between reference and system output span of their Arg-1/Arg-2 mention heads was required a positive overlap and " argument values are defined to be 1 if the arguments are mappable, 0 otherwise."[15]For the research of ACE systems were provided Source language data and evaluation (through an evaluation test corpus); training corpora was subdivided to include a development test set and training data were newly annotated. ACE05 training and evaluation data was selected by requiring a certain density of annotation across the corpus. All sources files provided in four versions were encoded in UTF-8 and only text between the begin text tag and end text tag were evaluated, with the exception of one: TIMEX2 annotation was placed between the and tags, notwithstanding that they occur outside the TEXT tags. The data format integrity (given in APF, AG and original source document format) was verified by three DTD’s[21]and a new evaluation data set was defined for the 2005 evaluation. The specification of entity mentions in terms of word locations in the source text became an essential part of system output, while word/phrase location information is given related to the indices of the first and last characters of the word/phrase, which ACE systems must compute them from the source data. The first character of a document received the index 0. Information and annotation provided as bracketed SGML tags are not counted while white-spaces and all the characters outside of angle-bracketed expressions were counted. Each new line (nl or cr/lf) was counted as one character. Each ACE target contributed to the score for each document that mentions that target, multiplying the score, and all tasks were scored using " document-level processing"[15]mode according to which each document is processed independently, all entities and relations mentioned in a single document were uniquely associated and identified with that document and it is not allowed nor required any reconciliation of ACE targets. Scores were reported over the entire evaluation test set and separately for each source domain, thus giving contrasts between different sources. The rules established for the participant systems were the following: changes of systems and human test data examination were not allowed after the evaluation date were released; all documents from all sources for every specific evaluation combination were processed for each submitted system output; every participating site submitted a detailed system description to NIST and attended the evaluation workshop. An XML validator[23]for verifying if the system output file conformed to ACE DTD and for validating their result along with the ACE evaluation software (which scored EDR, VAL, TERN, RDR, and VDR output) were available from NIST ACE web site. It was established the outlined step-by step procedure for submitting results which includes creating directory for each of the languages attempted (Arabic, Chinese or English), subdirectory for each task, containing one directory for each system submitted, for depositing all system output files in the appropriate system directory. The files of the results were compressed and transferred to NIST by FTP. System description was a valuable tool in discovering the strengths and weakness of different algorithmic approaches and in determining which sites needed oral workshop presentations or talks in a poster session. Each system description included: " the ACE tasks and languages processed; identification of the primary system for each task; a description of the system (algorithms, data, configuration) used to produce the system output; how contrastive systems differ from the primary system; a description of the resources required to process the test set, including CPU time and memory;"[15] and applicable references. NIST created a report which documents the evaluation and posted it on the NIST web space along with the list of participants and the official ACE value scores achieved for each task/language combination.

## Coreference task in MUC

The first five Message Understanding Conferences [24], focused only on " information extraction" task, requiring the free text analyzes, the specified type events identification, and the data base template’s filling with information about each such event. For the MUC-6 anonymous " dry run" evaluation, the tasks were: Named Entity Recognition (NER), which imply the recognition of entity names (for people and organizations), place names, temporal expressions, and certain types of numerical expressionsCoreference; which imply the identification of coreference relations among noun phrasesTemplate elements; which imply the extraction of information related only to a specified events’ class and the template’s filling for each such an event’s instance, andScenario templates (traditional information extraction). Two scenarios were released before the evaluation: the first one involving orders for aircraft, which was created using articles from the Wall Street Journal which were available in machine-readable form on the ACL/DCI disk, and distributed by the Linguistic Data Consortium andthe second one involving labor negotiations, which was used for the dry run. Discontinuous noun phrases appeared frequently in headlines in the MUC-6 corpus, since the non-first lines of a headline were often marked with "@", which pointed out that the non-first lines were external to the preceding and subsequent text. In MUC-7 Coreference Task [25], the coreference " layer" links together multiple expressions designating a given entity. The layer collected together all nouns’ mentions, including those tagged in the Named Entity task. The relations related to verbs were ignored. The coreference layer provided the input to the template element task. The criteria for the MUC-7 Coreference Task definition, in order of priority, were:" 1) Support for the MUC information extraction tasks; 2) Ability to achieve good (ca. 95%) interannotator agreement; 3) Ability to mark text up quickly (and therefore, cheaply); 4) Desire to create a corpus for research on coreference and discourse phenomena, independent of the MUC extraction task." [25]. The annotation scheme covers only the " IDENTITY" (or IDENT) relation for noun phrases, with no distinction between types, functions, and instances, because preserving high inter-annotator agreement is more important than capturing all phenomenons falling under the heading of " coreference". The task did not cover the coreference among clauses, or coreference relations (set/subset, part/whole, etc.)The IDENTITY (IDENT) relation is symmetrical, transitive and non-directional, thus inducing a set of equivalence classes among the marked elements. All elements are coreferring in an equivalence class. Each element participates in exactly one equivalence class. Where an expression may be coreferential with either of two NPs, a problem arose, " because of conjunction, or because of type/instance ambiguity or in expressions of change over time." [25] Two clearly distinct values /instances should were allowed to merge into an equivalence class, even if all of the function/value or type/instance relations were not marked. The coreference relation was assumed as being symmetric and transitive. The annotation contained the information establishing the type of link between an explicitly marked noun phrases’ pair, by SGML tagging within the text stream. Each string was marked up. Explicit links were used to infer the other links. The " TYPE" attribute indicated the relationship between the anaphor and the antecedent. Only " IDENT" relationship was being annotated. The ID and REF attributes were used to highlight the coreference link between two strings. During markup each ID is assigned arbitrarily and uniquely to the string. The REF used the ID to signal the coreference link. In the answer key (" key") was used the MIN attribute that showed the minimum string which the evaluated system must included in the COREF tag in order to receive full credit for its output (" response"). Valid responses included the MIN string and excluded all tokens beyond those enclosed in the ... tags. The phrase’s HEAD was in general the MIN string. When the markup was optional, in the answer key the STATUS (" status") attribute, having the only value OPT (" optional"), was used. The strings marked OPT in the key were not scored unless the response had markup on it. Only for the anaphor the optionality was marked. Coreference markup was made on the text’s body and on corpus- header’s specific portions. Various SGML tags were used to identify the body and the header’s various portions. The coreference’s annotation was carried our within the text delimited by the SLUG, DATE, NWORDS, PREAMBLE, TEXT, and TRAILER tags. The " erased" transcript’s portion, containing disfluencies or verbal erasures, were not annotated for coreference. Having the text annotated for disfluencies before beginning coreference annotation was helpful, establishing what was part of the final output. The coreference relation between: Nouns, including noun-like present participles preceded by an article or followed by an " of" phraseNouns Phrases such as an assertion’s object, a negation, a question or the initial introduction of an object ( including dates, currency expressions and percentages); andboth personal (all cases, including the possessive and the pronouns’ possessive forms used as determiners) and demonstrative Pronouns, were marked. The relation was marked only between pairs of markables elements. " Markable" element might not be followed by later references to it. Some looking anaphoric markables were not coded. Predicate nominals were typically coreferential with the subject. When the predicate nominative was marked indefinite, the coreference was recorded. An extensional descriptor was defined as a set members’ enumeration by (unique) names. Proper names and numerical values were an extensional descriptor in the coreference task. Indefinite appositional phrases and appositional phrases which constituted a separate noun phrase following the head, were markable; but negative appositional phrases were not markable. Appositional phrases were also marked in the specifier relation. No coreference was marked when a partial set overlap occurred. An intensional description was defined as a true predicate of an entity or set of entities which characterizes or defines the set’s members. All non-concrete common noun are alone intensional description: it functions at the " type" level or at the " function" level, if it takes a quantifiable value. For sets having no finite extension or without a known extension the intensional descriptions are useful. An intensional description could also be used regarding type’s instances, or function’s values. The grounding instance in a coreference chain was defined as the first extensional description in the chain. These terms were useful in the discussion regarding function-value relations, time-dependent entities and bare nominals. Losing some type coreference was allowed to prevent the collapsing of coreference chains. Named Entity’s substrings, pronouns without an antecedent or referring to a clausal construction, prenominal modifiers not being coreferential with a named entity or to the syntactic head of a maximal noun phrase, and relative pronoun were not markable. Names, date expressions and date’s components, gerunds and other clearly not decomposable identifiers were treated as atomic. In the coreference chain there must be one element, a head or a name, that is markable. The noun appearing at the head of a noun phrase is markable only as part of the entire noun phrase. The empty string is not markable. The MINimal string was defined as the span from the first " head" of the noun phrases having two or more heads through the last " head". The entire maximal conjoined noun phrase was included in the MAXimal string. The individual conjuncts, being separately coreferential with other phrases, are markable. In order to maximize the markables’ identifying, the system generated string must include the head of the markable, and may include additional texts up to a maximal NP. The maximal NP was enclosed in SGML tags in the key’s development. The MIN attribute assigned the NP’s head, which was mostly the main noun, without its left and right modifiers. The entire name assigned as head, including suffixes and excluding personal titles or any modifiers, was marked. Each name of multiple names location designators was considered a separate unit having generally the first of these names as the head. The other names were treated as first name’s modifiers. The minimal phrase used the syntactic head ignoring idioms or collocations constructions. When the head and the maximal noun phrase were the same or differ only by the articles " a" or " the", the MIN was not marked. All modifiers of the NP’s text such as appositional phrases, non-restrictive relative clauses, and prepositional phrases, were included in the maximal NP. The punctuation and the leading articles were striped by the scorer before comparing key to response and strings. The maximal noun phrase for the conjoined phrase having shared complements or modifiers was the maximal noun phrase. The minimal noun phrase span was between the first conjunct of the minimal phrase and the end of the minimal phrase for the last conjunct. Discontinuous noun phrases were included within a single COREF tag. In spoken languages’ transcripts, a noun phrase could be interrupted by a silence’s indication or by another speaker’s utterance. The MIN was not explicitly marked when the presence of an article such as " the", " a", or " an" at the NP’s beginning was the only difference between the head and the maximal NP. Two markables were linked when they were coreferential, by referring to the same object, set, activity, etc. A " bound anaphor" and the NP which binds it were coreferencial linked. A quantified NP was also linked through the coreference identity relation to subsequent anaphors, outside the quantification’s scope. Relative clauses bounded to the clauses’ head were coreferencial linked to the entire NP. For appositional phrases were provided alternative object’s descriptions or names. Other modifiers could separate appositional phrase from the head. Punctuation was generally not captured in text-to-speech transcription. Constructions looking similar to an appositive, but occuring within a single noun phrase as a title or modifier, were not considered markable. Two markables were recorded as coreferential when the text asserted them to be coreferential at any time. Coreference was marked for copula clearly implied by the verb’s semantics, equivalence’s expressions involving the word " as" and NPs enclosed in asterisks. Two markables both referring to sets/types, when the sets/types were identical, were coreferential. Most bare plurals’ appearances relate to types or kinds, not to sets. Phrases referring to the same amount of money are coreferential. When both extensions (extensional and intensional descriptions) are in the same clause, the function takes on the most " current" value in its clause. Coreference was determined with respect to coerced entities. No coercion was necessary for Countries, which are both geographical entities and governmental units, and their occurrences were coreferential. Any correct key compared to any correct response yield a 100% recall/100% precision score, regardless of how the coreference relation was encoded in the key by REF pointers.

## Coreference resolution as a graph problem

The pairwise coreference model (Soon et al., 2001, [3]) is based on an entity-mention graph, in which any two mentions belonging to the same equivalence class are connected by a path. The mentions’ contexts are the graph’s nodes. The pairwise coreference function, pc, is used to indicate the value of the probability that the two mentions are coreferential[3]. The coreference graph in the Bengtson and Roth’s research, (2008), [1], was generated using the Best-Link decision model (Ng and Cardie, 2002b, [26]). Each component connected in the graphrepresents one equivalence class. Some links between mentions are detected without knowing whether other mentions are linked. These mentions’ equivalence classes are determined through the transitive closure of all links. Pronouns are not considered as candidate antecedents when the mention is not a pronoun. In the Bengtson and Roth’s research [1]the official ACE 2004 English training data (NIST, 2004, [27]) was used for the experimental study. The ACE 2004 corpus was split into three sets: Train , which contains a random 80% of the 336 documents in their training setDev, which contains the remaining 20% of the 336 documents in their training set, andTest, which contains the same 107 documents as Culotta et al. (2007, [28]). For the ablation study the development set was randomly split into two parts: Dev-Tune, to optimize B-Cubed F-Score, and Dev-Eval. In all experiments words and sentences were automatically split using the given pre-processing tools [29]. The document-level pairwise coreference model included the following constraints: non-pronouns cannot refer back to pronouns, andas training examples were used all ordered pairs of mentions, subject to the above constraintThe pc quality depends on the used features. For each mention mthe closest preceding mention awas selected from m’s equivalence class and the pair (a; m) was presented as a positive training example, assuming that the existence of this edge is the most probable. For all mentions a that precede m and are not in the same equivalence class were generated the negative examples (a; m). In the Bengtson and Roth’s research [1] were used the Boolean features and the conjunctions of all pairs of features. The mention type pair feature was used in all experiments to indicate whether the mention is a proper noun, a common noun, or a pronoun. String relation features were used to indicate whether strings are sharing some property or both are sharing a modifier word. Only modifiers occurring before the head were taken into account. Semantic Features were used to determine the matching of gender or number, or whether the mentions are synonyms, antonyms, or hypernyms, and to check the relationship of modifiers that share a hypernym. Gender was determined by the existence of mr, ms, mrs, or by the gender of the first name, or ends of organization’s name (inc., etc.), or proper names. A gender is assigned to common nouns by classifying it as male, female, person, artifact, location, or group from the hypernym tree, using WordNet. A, an, or this indicates the singular; those, these, or some indicates plural. Two mentions having the same spelling are assumed to have the same number. WordNet Features were used to check whether any sense of one head noun phrase is a synonym, antonym, or hypernym, or any sense of the phrases share a hypernym. Modifiers Match was used to determine whether the text before the mention’s head matches the head, while Both Mentions Speak was used as a proxy for having similar semantic types. Relative Location Features was used to measure distance for all i up to the distance and less than some maximum, for the mentions being in the same sentence. The number of compatible mentions was used as a distance. The mentions separated by a comma are appositions. Mentions having the same gender and number are compatible. A basic trained classifier was used for finding coreferential modifiers. A separate classifier, which predicts anaphoricity with about 82% accuracy, detected anaphoric mentions, which became features for the coreference model. The relationship (match, substring, synonyms, hypernyms, antonyms, or mismatch) of any modifiers’ pair that shares a hypernym was determined. Modifiers were restricted to single nouns and adjectives occurring before the head noun phrase. The presence or absence of each pair of final head nouns, one from each example mention, was treated as a memorization feature. The entity type (person, organization, geo-political entity, location, facility, weapon, or vehicle) was predicted using lists of personal first names, of honorary titles, of personal last names drawn from US census data, of cities, states, countries, organizations, corporations, sports teams, universities, political parties, and organization endings. The checking return unknown when the name appears in more than one list. Common nouns were checked with the hypernym tree. The entity is recognized as a person only for personal pronouns. Entity Type Match feature was used to verify if the predicted entity types match, and return " true if the types are identical, false if they are different, and unknown if at least one type is unknown." [1]. Boolean Entity Type Conjunctions feature was used to indicate the presence of the pair of predicted entity and mention types for the two mentions by replacing the type in the pair with the word token. Anaphoricity was used as a feature for a learning algorithm. For training a threshold of 0. 0 was used, the learning rate was 0. 1 and the regularization parameter was 3. 5. The number of training rounds was allowed from 1 to 20. The parameters were chosen to optimize B-Cubed F-Score when evaluating. The term end-to-end-coreference was used for a system able to determine the coreference on plain text. The Bengtson and Roth’s research [1] stages were: The detection of mention heads using standard featuresThe detection of each head’s extent boundaries using a learned classifierThe establishing of mention’s type using a learned mention type classifierThe application of the above described coreference algorithm

## Coreference Resolution on Blogs and Commented News

Iris Hendrickx and Veronique Host studied the effect of the genre shift from edited structured newspaper text (newspaper articles, mixed newspaper articles and reader comments, and blog data), to unedited, unstructured blog data [30]. Coreferential resolution on blogs and news articles with user comments is involved in the identification of the opinion holder (the person, institution, etc. that holds a specific opinion on a particular object) and the target (the subject/topic of the expressed opinion). The data sets, used for their coreference system’s comparison, were: newspaper articles from the KNACK 2002 data set, containing 267 manually annotated Dutch news articles no longer than 20 sentences, produced by professional writers5 mixed manually annotated newspaper articles and reader comments, each having an author and time stamp, andmanually annotated blog data, from 15 blog posts from two different blogs, containing interactive diary entries about a certain event/textson events in Belgian cities, written by multiple authors. The majority of the reader comments, which were from 88 to 123 per article, contained up to two short sentences. As the comments refer to the entities mentioned in the news article, to simplify, each news article and the accompanying reader comments were treated as one single document. The type and quantity of anaphors in the test sets confirm that the blogs and commented news both contain relatively more pronouns than the newspaper articles. Each pair of noun phrases in these chosen texts were classified as having a coreferential relation or not, and a feature vector, which denotes each pair of noun phrases’ characteristics and their relation, was created. The creating process of the feature vectorsincludedthe following steps: a rule-based system using regular expressions executed the tokenization. the memory-based tagger MBT carried out the part-of-speech tagging and text chunking, andthe memory-based relation finder searched for the grammatical relations between chunks in order to establish the subject, object, etc. An automatic Named Entity Recognition system was used also for the task. In addition, to refine the predicted label person to female or male, a lookup for names in gazetteer lists was fulfilled by the system. World knowledge along with a combination of information from morphological, lexical, syntactic, semantic and positional sources, were also used. The string overlap, overlap in grammatical role and named entity type, synonym/hypernym relation lookup in WordNet, distance between the noun phrases, morphological suffix information and local context of each of the noun phrases were extracted. For pronouns, named entities and common nouns were created three separate systems which were used to separately and iteratively optimize the machine learning classifier having the implemented learning algorithm of the software package Timbl. 242 articles were used for training and 25 articles, along with the entire blog data set and news comments data set were used for testing. The precision, recall and F-score were measured using the MUC scoring software and the recall was computed with B-Cubed method.

## Beautiful Soup Package

Beautiful Soup [31] is a Python library allowing different parsing strategies or trade speed for flexibility, and a toolkit for analysing HTML and XML files, for extracting parts of their content and also for reconstructing the initial parse of the document. The package provides methods and Pythonic idioms for navigating, searching, and modifying parse trees, and is used for automatically converting incoming documents to Unicode and outgoing documents to UTF-8. Beautiful Soup auto-detects the usual encodings. The HTML parser from Python’s standard library and Python parsers such as lxml parser are supported by Beautiful Soup. Python’s html. parser is lenient and has a decent speed, lxml’s HTML parser is lenient and very fast, lxml’s XML parser is the only currently supported XML parser and very fast, while html5lib is extremely lenient, creates valid HTML5, parses pages in the same manner as a web browser does. An HTML parser takes the string of characters and turns it into a series of events. A BeautifulSoup object, obtained by running the HTML/XML files through Beautiful Soup, represents the document as a nested data, as a whole. BeautifulSoup object can be treated as a Tag object, but it has no name and no attributes. Common tasks are: extracting all the text from a pageThe BeautifulSoup constructor is used to parse the documents, first by converting documents to Unicode and HTML entities to Unicode characters, and then by using the best available parser. The document is transformed into a complex tree of Python objects, classified as follows: Tag objects, which corresponds to an XML or HTML tag in the original documentName, every tag has a name, accessible as . nametag’s Attributes, multi-valued attribute, as a tag can have more than one CSS class, rel, rev, accept-charset, headers, and accesskeyTag’s name and attributes are its most important features. Tags may contain strings and other tags, which are the tag’s children. All HTML mark-up generated by Beautiful Soup reflect the tag’s name change. Tag’s attributes are accessed, added, removed, and modified by treating the tag like a dictionary or are directly accessed as . attrs. The value(s) of a multi-valued attribute is presented as a list. Turning tags back into strings consolidates multiple attribute values. NavigableString class contains bits of text corresponding to each string. NavigableString can be converted to a Unicode string with unicode(). Strings can be replaced one with another using replace\_with(). Strings don’t support the . contents, or . string attributes, or the find() method. Tag, NavigableString and BeautifulSoup cover the HTML or XML file’s content with a few exceptions, such as the comment. The Comment object is a special type of NavigableString, displayed with special formatting. Other subclasses of NavigableString are: CData, ProcessingInstruction, Declaration, and DoctypeTo navigate between page elements being on the same parse tree’s level are used . next\_sibling and . previous\_sibling. The . next\_element string’s or tag’s attribute shows whatever was parsed immediately afterwards, while the . previous\_element attribute indicates whatever element was parsed immediately before. Methods for searching the parse tree and for isolating parts of the document are find() and find\_all(name, attrs, recursive, text, limit, \*\*kwargs), which looks through a tag’s descendants and retrieves all descendants that match your filters. Filters are based on a tag’s name, on its attributes, on the text of a string, or on some combination of these. The output of using a string as the simplest filter through Beautiful Soup will be a match against that exact string. A regular expression object is filtered in Beautiful Soup by using its match() method. Beautiful Soup allows a string match against any item in a given list. The value True is matching everything it can. Defined function, which takes an element as its only argument, should return True if the argument matches, and False otherwise. An attribute can be filtered based on a string, a regular expression, a list, a function, or the value True. All unrecognized arguments are turned into filters on one of a tag’s attributes. Multiple attributes are filtered at once by passing in more than one keyword argument. In Python the CSS attribute’s name " class" is a reserved word. To search by CSS class is used the keyword argument class\_ a string, a regular expression, a function, or True. A single tag can have multiple values for its " class" attribute. For search strings is used text. find\_all() returns all the tags and strings that match the given filters. To stop gathering results, after it’s found a certain number of tags and strings, is used a number for limit, which works just like the LIMIT keyword in SQL. To consider only direct children is used recursive= False, in find\_all() and find() methods. find\_all() and find() work looking at tag’s descendants, while find\_parents() and find\_parent()work looking at a tag’s (or a string’s) parents. The find\_next\_siblings()and find\_previous\_siblings() methods return all the siblings that match, while find\_next\_sibling()and find\_previous\_sibling()only return the first one that match. The find\_all\_next() and find\_all\_previous() methods return all matches, while find\_next()and find\_previous() only returns the first match. A subset of the CSS selector standard is supported by Beautiful Soup. The tree can be changed using Beautiful Soup and the changes will be written as a new HTML or XML document. Tag names and attributes can also be changed. To add a string to a document is used append(), or BeautifulSoup. new\_string(). To create a whole new tag is used BeautifulSoup. new\_tag(). Beautiful Soup methods also allows to append, extract, insert, clear, decompose or replace a tag. Using Beautiful Soup methods a specified element in the tag can be wrapped or unwrapped. A Beautiful Soup parse tree is turned into a nicely formatted Unicode string, having each HTML/XML tag on its own line, by using the prettify() method. unicode() or str(), or a Tag within it, used on a BeautifulSoup object returns a string without a special formatting. HTML entities like"&lquot;" contained in a document will be converted by Beautiful Soup to Unicode characters. After converting the document to a string, the Unicode characters will be encoded as UTF-8, and the HTML entities will not be recovered. Bare ampersands and angle brackets are the only characters that are escaped upon output, being turned into "&", "<", and ">". This behavior can be changed by providing a value for the formatter argument to prettify(), encode(), or decode(). Beautiful Soup recognizes for formatter the values: minimal, html or none. The EntitySubstitution class in the bs4. dammit module implements Beautiful Soup’s standard formatters as class methods. The text inside a CData object is always presented exactly as it appears, without formatting. The get\_text() method returns all the text in a document or beneath a tag, as a single Unicode string. Beautiful Soup ranks lxml’s parser as being the best, followed by html5lib’s, and Python’s built-in parser. The order is overridden by specifying the preferred type of markup and the installed parser library’s name. Beautiful Soup presents the same interface to a number of different parsers which will create different parse trees from the same document. Any HTML or XML document, which is written in a specific encoding like ASCII or UTF-8, is converted to Unicode by Beautiful Soup, and the original encoding remains available as the . original\_encoding BeautifulSoup object’s attribute. All writed out documents from Beautiful Soup are UTF-8 documents. prettify() can change the writed out documents’ encoding. Unicode, Dammit detects document’s encodings converting them to Unicode, and converts Microsoft smart quotes to HTML or XML entities. UnicodeDammit. detwingle() is used to turn inconsistent encodings into pureUTF-8, before passing the document into BeautifulSoup or the UnicodeDammit constructor. The SoupStrainer class allows users to choose which incoming document parts are parsed, except for html5lib parser. The SoupStrainer class arguments are name, attrs, text, and \*\*kwargs. Beautiful Soup parses documents as HTML by default. Using lxml as the underlying parser can speed up Beautiful Soup. Parsing only part of a document can save memory, and increase the speed of the document’s searching.

## Evaluation metrics

In previous coreference evaluations were used different evaluation metrics, which could not produce the same ranking of the systems. Choosing the best fit metrics for the task is difficult because " each metric generates its variation of a precision and recall measure."[13]The MUC measure focuses on the links in the data and is the most widely used and is based on the minimal number of missing and wrong links, taking into account only coreference links, determined as the total number of mentions minus the number of entities. The recall (REC) is determined as the number of common links between entities in K and R divided by the number of links in K, and precision (PER) is the number of common links between entities in K and R divided by the number of links in R, where K represent a set of key entities comprising one or more mentions, and R a set of response entities.

## ()

and

## ()

F-measure is a trade-off between REC and PRE. Pairwise F1 computes PER, REC and F over all pairs of coreferent mentions, ignoring the correct identification of singletons.

## ()

This metric is used for systems having more mentions per entity, because it ignores recall for singleton entities. In the Bengtson and Roth’s research [1] , the pairwise coreference function, pc, was used to indicate the value of the probability that the two mentions are coreferential The highest scoring antecedent, a, was determined as follows:

## ()

wherem represent a mention, and Bm is the set of mentions appearing before m in the document. The edge (a; m) was added to the coreference graph Gd if pc(a; m) is above a threshold. The pairwise coreference function pc takes as input ordered pairs of mentions (a; m) such as a precedes m in the document, and produces as output a value representing the conditional probability that m and a belong in the same equivalence class. The MUC F-score [1] is the harmonic mean of precision, which computes the minimumnumber of added links ensuring that all mentions referring to a given entity are connected in the graph, and recall, which computes the number of removed links ensuring that no two mentions referring to different entities are connected in the graph. The B-CUBBED metric (Bagga and Baldwin, 1998, [31])is based on the mentions including singletons and computes recall and precision scores for each mention. As singletons are the largest group in real texts, B-CUBBED scores rapidly approach 100%, thus obscuring the level of accuracy of a system in terms of coreference links. The recall (REC) is determined as the total number of intersecting mentions between K and R divided by the number of links in K, and precision (PER) is the number of common links between entities in K and R divided by the total number of mentions in R, where K represent a set of key entities comprising one or more mentions, and R a set of response entities. The average over the individual mention scores gives the final scores.

## ()

and

## ()

B-Cubed F-Score [1] is the measure of predicted clusters and true clusters overlap, computed as the harmonic mean of recall:

## ()

and precision:

## ()

where nm is the number of mentions appearing both in m’s predicted and true clusters, st is the size of m’s true cluster, while sp is the size of the m’s predicted cluster; d isone of the document’s set D, and N is the total number of mentions in D. B-Cubed F-Scoregives equal weight to all types of entities and mentions. Constrained Entity Aligned F-measure (CEAF) is based on entities, aligning each response entity with at most one key entity using an entity similarity metric (Luo, 2005,[32] ). CEAFe is based on entity and CEAFm is based on mention. The CEAF precision and recall are derived from the alignment which has the best total similarity (denoted as Φ(g∗)). Recall is the total similarity divided by the number of mentions in K, and precision is the total similarity divided by the number of mentions in R.

## ()

and

## ()

BiLateral Assessment of Noun-Phrase Coreference (BLANC) uses avariation on the Rand index (Rand, 1971, [33]) suitable for evaluating coreference (Recasens and Hovy, 2011, [34]). The Rand index equals the number of mention pairs that are either placed in an entity or assigned to separate entities in both K and R, normalized bythe total number of mention pairs in each partition. In different entities, in both K and R, the mention pairs’ number is high. BLANC rewards correct entities according to the number of mentions but assume that the sum of all coreferential and non-coreferential links is constant for a given set of mentions, implying identical mentions in key and response. MELA metric (Denis and Baldridge, 2009,[35] )uses a weighted average of three metrics: MUC, B- CUBED, and CEAF. ACE-Value (Doddington et al. 2004, [11]) is task-specific, being limited to a set of specific semantic types. The score is computed by subtracting the normalized cost, which corresponds to the sum of errors produced by unmapped and missing mentions/entities and wrong mentions/entities, from 1. The cost associated to each error was changed between successive evaluations.

## Comparative results

The difficulty in comparing the performance of different coreference resolution systems is compounded by factors, such as: the use of different test sets (Stoyanov et all., 2009,[36]); the use of true or system mentions; the evolution of evaluation criteria; the evolution both in size and scope of training and/or evaluating data sets; the troublesome detection of heavily nested mentions; andthe mismatches in mention boundaries, or missing mentions which affect the systems’ evaluation against a gold standard corpus. The typical errors made by the resolution systems are: Pronouns that have no clear antecedent (especially this & that, as well as pleonastic pronouns and personal pronouns, which are often used in connection with people in general and not to a specific entity mentioned in the text, were linked with a preceding NP)were erroneously classified as being coreferential; Noun phrases were partially detected, fact that induce a partial coreferential relations’ recognition ; The feature vector might not provide clear information to differentiate the positive classification from the negative one; Some ambiguities, such as abbreviations, required' world knowledge' or sophisticated information resources; Some algorithms produced broken/ glued/ misplaced entity, which affect entity resolution’ scoreSome mentions were considered generic mention, which affect mentions resolution’ scoreMost systems have removed singletons from the response, so they were not credited for the correctly removed from the data singleton entities, and were penalized for the accidentally linked with another mention singletons;

## Conclusions on previous coreference tasks

There are two main research trends in the field of coreference resolution: the first one is to incorporate more features into the models, andthe second one is shifting from the supervised learning setting to unsupervised setting. Features such as mention-pair model and cluster ranking models are coming from FrameNet, WordNet, Yago or other Semantic knowledge sources, also from Annotated Corpus, such as ACE, MUC, or OntoNotes, as well as from Unannotated Corpus, represented by horn rules or just relations. Rule-based systems outperformed trained systems. The features for coreference prediction are complex, and it is difficult to combine them in order to obtain the best results. The coreference performance was not improved by using gold standard information on the other individual layers. Merging various attributes of the mentions’ information was useful for improving only some systems’ performance. Coreference, anaphoricity, and named entity classification's joint inference using Integer Linear Programming (ILP) have improved the main coreference metrics scores (MUC, B-CUBBED, and CEAF). Most systems have a filter to avoid double-credited duplicate mentions. Mention boundary errors did not contribute significantly to the final output and the gold mentions data have improved the system’s performance. The head word based scoring relaxed strategy did not much improve the system’s performance for the open and closed track submissions. Systems’ performance did not seem to vary much across the different genres. For unedited texts, such as the blog and news with comments test sets, the coreference resolution systems’ performance dropped significantly. An expanded and improved data set consisting of spoken language and annotated blogs and commented news along with factual information gathered from the all available sources are necessary both for training and for testing coreferential systems. In question-answering systems the regularly applied method is the facts’ finding. The information related to turn taking in dialogs might also be exploited in blogs and reader comments’ coreferential resolution. State-of-the-art systems with complex models can be outperformed by a pairwise model with strong features. Some features such as aligned modifiers or relative pronoun feature contribute more to precision, while others, such as learned features or apposition features, serve more to recall. For systems making links based on the best pairwise coreference value the distance features are important. As two coreferential mentions cannot have different entity types, the Predicted Entity Types feature contribute with its selective power. A body of general principles to codify the rules for coercion and coreference should be developed. The coreference judgments should be based on the intelligent reader's best understanding of the text.

## Stanford coresolution tools

Stanford CoreNLP is an integrated framework providing model files for analysis of English along with a set of natural language analysis tools, including " the part-of-speech (POS) tagger, the named entity recognizer (NER), the parser, and the coreference resolution system. "[37]Raw English language plain text is analysed running the tools provided by Stanford CoreNLP. The analyses’ output are the base forms of words, their parts of speech, whether they are names, dates, times, or numeric quantities, and the sentences structure marking up, indicating which noun phrases refer to the same entities. This kind of text analyses is useful for further higher-level and domain-specific text understanding applications.

## Stanford Deterministic Coreference Resolution System

The top ranked system at the CoNLL-2011 shared task, described in Section 2. 1. 2., was the Stanford Deterministic Coreference Resolution System. The " system implements the multi-pass sieve coreference resolution (or anaphora resolution) system described in Lee et al. (CoNLL Shared Task 2011, [38]) and Raghunathan et al. (EMNLP 2010, [39]). The score is higher than that in EMNLP 2010 paper because of additional sieves and better rules (see Lee et al. 2011 for details). Mention detection is included in the package." [40]

## Stanford Named Entity Recognizer (NER)

Stanford Named Entity Recognition (NER or CRF Classifier) is a Java implementation of a Named Entity Recognizer, which labels sequences of words / names of things in a text. The Stanford NER software " provides a general (arbitrary order) implementation of linear chain Conditional Random Field (CRF) sequence models, coupled with well-engineered feature extractors for Named Entity Recognition." [41] With the Stanford NER are available for download3 class (Location, Person, Organization) model trained on both data sets for the intersection of those class sets4 class (Location, Person, Organization, Misc) model trained for CoNLL, and/or7 class (Location, Person, Organization, Money, Percent, Date, Time, ) model trained for MUC, named entity recognizers for English, and a pair of models trained on the CoNLL 2003 English training data. Each of these models use distributional similarity features, which increase both system’s performance and models’ size and runtime, and require more memory. The same models are also available without those features. As part of a caseless models’ package, are available the same three models’ caseless versions. Stanford NER, licensed under the GNU General Public License, with the source, is available for download. The provided software adds new distributional similarity based features. The models, trained on a mixture of CoNLL, MUC-6, MUC-7 and ACE named entity corpora, are fairly robust across domains. " The package includes components for command-line invocation, running as a server, and a Java API."

## Training a NER model

Basically, the training data are in tab-separated columns. The meaning of the columns is defined via a map. The column containing the NER class, and existing features know about names like " word" and " tag" is called " Answer". The data file, the map, and what features to generate are defined through a properties file. The file containing the text for training is converted to one token per line with Stanford tokenizer (which was included in the box). Various annotation tools are available to make training data by labelling the entities. The labelling can also be made by hand in a text editor. The default label is " O" in Stanford software The label can also be specified through the backgroundSymbol property and then the real entities are hand-labelled in a text editor. Through labelling only one entity type, PERS for person name, was marked, but a second entity type such as LOC for location, can be easily added to this data. To see how well the system is doing, some test data contained in another file are necessary. Using a properties file is easier than specifying all properties on the command line in Stanford NER CRF. Once the program has completed the NER model will be serialized to the location specified in the properties file (ner-model. ser. gz). The input tokens, the correct (gold) answers, and the answer guessed by the classifier are the columns of the output. By looking at the output, you can see that the classifier finds most of the person named entities but not all, mainly due to the very small size of the training data (but also this is a fairly basic feature set). The code then evaluates the performance of the classifier for entity level precision, recall, and F1. It gets 80. 95% F1. (A commonly used script for NER evaluation is the Perl script conlleval, but either it needs adapted or else the raw IO input format used here needs to be mapped to IOB encoding for it to work correctly and give the same answer.)So how do you apply this to make your own non-example NER model? You need 1) a training data source, 2) a properties file specifying the features you want to use, and (optional, but nice) 3) a test file to see how you're doing. For the training data source, you need each word to be on a separate line and annotated with the correct answer; all columns must be tab-separated. If you want to explicitly specify more features for the word, you can add these in the file in a new column and then put the appropriate structure of your file in the map line in the properties file. For example, if you added a third column to your data with a new feature, you might write " map= word= 0, answer= 1, mySpecialFeature= 2". Right now, most arbitrarily named features (like mySpecialFeature) will not work without making modifications to the source code. To see which features can already be attached to a CoreLabel, look at edu. stanford. nlp. ling. AnnotationLookup. There is a table which creates a mapping between key and annotation type. For example, if you search in this file for LEMMA\_KEY, you will see that lemma produces a LemmaAnnotation. If you have added a new annotation, you can add its type to this table, or you can use one of the known names that already work, like tag, lemma, chunk, web. If you modify AnnotationLookup, you need to read the data from the column, translate it to the desired object type, and attach it to the CoreLabel using a CoreAnnotation. Quite a few CoreAnnotations are provided in the class appropriately called CoreAnnotations. If the particular one you are looking for is not present, you can add a new subclass by using one of the existing CoreAnnotations as an example. If the feature you attached to the CoreLabel is not already used as a feature in NERFeatureFactory, you will need to add code that extracts the feature from the CoreLabel and adds it to the feature set. Bear in mind that features must have unique names, or they will conflict with existing features, which is why we add markers such as "-GENIA", "-PGENIA", and "-NGENIA" to our features. As long as you choose a unique marker, the feature itself can be any string followed by its marker and will not conflict with any existing features. Processing is done using a bag of features model, with all of the features mixed together, which is why it is important to not have any name conflicts. Once you've annotated your data, you make a properties file with the features you want. You can use the example properties file, and refer to the NERFeatureFactory for more possible features. Finally, you can test on your annotated test data as shown above or annotate more text using the -textFile command rather than -testFile. Here are some tips on memory usage for CRFClassifier: Ultimately, if you have tons of features and lots of classes, you need to have lots of memory to train a CRFClassifier. We frequently train models that require several gigabytes of RAM and are used to typing java -mx4g. You can decrease the memory of the limited-memory quasi-Newton optimizer (L-BFGS). The optimizer maintains a number of past guesses which are used to approximate the Hessian. Having more guesses makes the estimate more accurate, and optimization is faster, but the memory used by the system during optimization is linear in the number of guesses. This is specified by the parameter qnSize. The default is 25. Using 10 is perfectly adequate. If you're short of memory, things will still work with much smaller values, even just a value of 2. Use the flag saveFeatureIndexToDisk = true. The feature names aren't actually needed while the core model estimation (optimization) code is run. This option saves them to a file before the optimizer runs, enabling the memory they use to be freed, and then loads the feature index from disk after optimization is finished. Decrease the order of the CRF. We usually use just first order CRFs (maxLeft= 1 and no features that refer to the answer class more than one away - it's okay to refer to word features any distance away). While the code supports arbitrary order CRFs, building second, third, or fourth order CRFs will greatly increase memory usage and normally isn't necessary. Remember: maxLeft refers to the size of the class contexts that your features use (that is, it is one smaller than the clique size). A first order CRF can still look arbitrarily far to the left or right to get information about the observed data context. Decrease the number of features generated. To see all the features generated, you can set the property printFeatures to true. CRFClassifier will then write (potentially huge) files in the current directory listing the features generated for each token position. Options that generate huge numbers of features include useWordPairs and useNGrams when maxNGramLeng is a large number. Decrease the number of classes in your model. This may or may not be possible, depending on what your modeling requirements are. But time complexity is proportional to the number of classes raised to the clique size. Use the flag useObservedSequencesOnly= true. This makes it so that you can only label adjacent words with label sequences that were seen next to each other in the training data. For some kinds of data this actually gives better accuracy, for other kinds it is worse. But unless the label sequence patterns are dense, it will reduce your memory usage. Of course, shrinking the amount of training data will also reduce the memory needed, but isn't very desirable if you're trying to train the best classifier. You might consider throwing out sentences with no entities in them, though. If you're concerned about runtime memory usage, some of the above items still apply (number of features and classes, useObservedSequencesOnly, and order of the CRF), but in addition, you can use the flag featureDiffThresh, for example featureDiffThresh= 0. 05. In training, CRFClassifier will train one model, drop all the features with weight (absolute value) beneath the given threshold, and then train a second model. Training thus takes longer, but the resulting model is smaller and faster at runtime, and usually has very similar performance for a reasonable threshold such as 0. 05.

## How do I train one model from multiple files?

Instead of setting the trainFile property or flag, set the trainFileList property or flag. Use a comma separated list of files.

## What options are available for formatting the output of the classifier?

Several options are available from the command-line for determining the output format of the classifier. You can choose an outputFormat of xml, inlineXML, or slashTags (the default). See the example of each below (these are bash shell command lines, the last bit of which suppresses message printing, so you can see just the output). Even more power is available if you are using the API. The classifier. classifyToString(String text, String outputFormat, boolean preserveSpaces) method supports 6 output styles (of which 3 are available with the outputFormat property: the XML output options preserve spaces, but the slash tags one doesn't). Even more flexibility can be obtained by using other of the classify.\* methods in the API. These will return classified versions of the input, which you can print out however your heart desires! There are also methods like classifyToCharacterOffsets(String) which returns just the entity spans. See the examples in NERDemo. java.

##  Is the NER deterministic? Why do the results change for the same data?

Yes, the underlying CRF is deterministic. If you apply the NER to the same sentence more than once, though, it is possible to get different answers the second time. The reason for this is the NER remembers whether it has seen a word in lowercase form before. The exact way this is used as a feature is in the word shape feature, which treats words such as " Brown" differently if it has or has not seen " brown" as a lowercase word before. If it has, the word shape will be " Initial upper, have seen all lowercase", and if it has not, the word shape will be " Initial upper, have not seen all lowercase".

##  How can I tag already tokenized text?

Use the following options:-tokenizerFactory edu. stanford. nlp. process. WhitespaceTokenizer -tokenizerOptions " tokenizeNLs= true"

##  Does the NER use part-of-speech tags?

None of our current models use pos tags by default. This is largely because the features used by the Stanford POS tagger are very similar to those used in the NER system, so there is very little benefit to using POS tags. However, it certainly is possible to train new models which do use POS tags. The training data would need to have an extra column with the tag information, and you would then add tag= X to the map parameter.

##  How do you use gazettes with Stanford NER?

None of the models we release were trained with gazette features turned on. However, it is not difficult to train a new model which does use the gazette features. Set the useGazette parameter to true and set gazette to the file you want to use. You need to supply the gazette at training time, and then the NER will learn features based on words in that gazette. The gazette will be included in the model, so does not need to be redistributed or given at test time. Any gazettes supplied at test time will be treated as additional information to be included in the gazette. There are two different ways for gazette features to match. The first is exact match, which means that if John Bauer is a gazette entry, the feature will only fire when John Bauer is encountered in the text. This is turned on with the cleanGazette option is set to true. The other is partial match, which is turned on with the sloppyGazette option. In that case, features fire whenever any of the words match, so John by itself would also trigger a feature in this case. The gazette files should be of the formatCLASS1 this is an exampleCLASS2 this is another example

## ...

## ...

## Beautiful Soup

## Summary

This chapter presents a literature review on both Stanford coresolution tool and other coresolution tools. In coresolution tools, several basic concepts including .............. are delved into. ............... Details with regard to a number of coresolution tools, namely ...., are covered. ExperimentsThis chapter investigates the effectiveness of Stanford coresolution tool, which is ...... as explained in Section Error: Reference source not found, against coresolution tool. .. purpose-.............. Stanford coresolution tool is used to............... The results obtained serve to reveal weaknesses and shortcomings in Stanford coresolution tool that this thesis intends to highlight and provide a solution for....

## ...

## ...

## ....

## ...

## ...

## ....

## ...

## ...

## ....

## ...

## ...

## Simulation Results

## ....

## ...

## ....

## ...

## ...

## ...

## ...

## ...

## ...

## Summary

## ...

DISCUSSIONS

## ....

## ....

## ....

## ....

## ...

## Summary

## ...............

Conclusions

## .....

Future Work

## ...

## .