# INFORMATION-BASED EVENT COREFERENCE

BY

RUICHEN WANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Professor Dan Roth

# Abstract

Event Coreference is an important module in the event extraction task, which has been shown to be difficult to solve. The goal is to link mentions talking about the same event together so that the information could be aggregated. This task could further be split into two slightly different subtasks: Within-Doc Event Coreference and Cross-Doc Event Coreference. Most of the related publications tried to solve the problem of Event Coreference in a two-step manner: Train or design a similarity metric for event mention pairs, then apply some clustering algorithm to the event mention space using the similarity metric as distance.

In this work, we identify two major problems people have neglected: One is that coreference does not imply full event mention similarity due to the fact that event mentions tend to contain partial and even complementary information. The other problem is that the order to compare event mentions pair could be important, because instead of comparing event mentions pairs that have incomplete and trustless information, comparing those who have complete and trustworthy information first could prune the error rate. We propose Core Similarity, a new argument-based similarity metric, to solve the first problem, and two information-based clustering algorithms for the second problem - Informative-First Clustering (IFC) for within-doc situation and Topic-Side Event Clustering (TSEC) for cross-doc situation. These clustering algorithms are based on the idea of Event Information which is defined in this work. Finally, the EVCO system is delivered with all of these details implemented.

*To my adviser, who backed me up when life was hard.*

# Acknowledgments

This work is supported by NSF grant #SMA 12-09359.

# Table of Contents

# Chapter 1

# Introduction

We are living in a Big Data era where information flows all around us. Most of the information is communicated and stored in the form of text. Thus, it is increasingly more necessary to be able to automatically extract information from extremely large amounts of text.

Event Extraction technique is one of the responses to this problem. Typically, Event Extraction models will take a piece of text (news articles, communication logs, etc.) as input, then output the events contained within the text. People will no longer need to read the full article to understand what is happenning, but just look at the processed results to know everything about the event. Specifically, an event could typically be defined with these event components: time, location, subject, and zero to two objects depending on different types of events.

Event Coreference could be seen as one of the final steps in the Event Extraction pipeline, which mainly tries to aggragate and deduplicate information related to events. There always exist several event mentions within a piece of text that are talking about the same event. The task of Event Coreference is to put these mentions together. This is not an easy task. Even human beings could not do it well - the interagreement between annotators on event related data is low (which could be considered as a fuzzy ceiling of automatic event coreference performance). Additionally, there is little literature on Event Coreference.

Taking a closer look, the problem could be divided into Within-Doc Event Coreference and Cross-Doc Event Coreference. Normally, coreferencing event mentions within the same document tend to contain insufficient but complementary information. Thus, by doing event coreference within document, we are actually aggregating information for a certain event so that we can get a complete summary of it. After that, we are able to do Cross-Doc Event Coreference by comparing the complete summaries. We can expect that, Within-Doc Event Coreference is step-by-step chaining event mentions together according to certain anchors (e.g. triggers, arguments, etc.), while Cross-Doc Event Coreference is simpler clustering algorithm provided complete information of events. Another way to see this concept: Within-Doc Event Coreference is clustering semantically-related (not necessarily similar) event mentions, while Cross-Doc Event Coreference is clustering semantically-similar events, which is merging similar clusters. We will have formal description about differences between these tasks after a survey of all related works and data.

Before going into details, we need to agree on some terms used throughout this thesis. We use ”event mention” to refer to an actual surface string containing event(s) in the text, while using ”event” to refer to the aggregation and summarization of coreferred event mentions.

In Chapter 2, we present a literature review on most publications on the topic of Event Coreference. After understanding the related literatures, we give a formal problem definition for Event Coreference and discuss typical solutions and their drawbacks in Chapter 3, which is concluded with the main proposed solution in this thesis. In Chapter 4, we introduce an important definition of Event Information to our thesis. In Chapter 5, we talk about the similarity metric used in our system, which is a little different from typical similarity metrics. In Chapter 6 and 7, we discuss the clustering algorithms proposed to solve both Within-Doc Event Coreference and Cross-Doc Event Coreference. Our working system is described in Chapter 8. Chapter 9 shows the experiments and the results. Chapter 10 discusses possible future works. Finally, Chapter 11 concludes this thesis.

# Chapter 2

# Survey

The problem of event coreference could be divided into two directions: within-document event coreference and cross-document event coreference. There are not sufficient work presented for the task of event coreference because it is not an easy task. We will go through all the present works in the following sections. After the literature reviews, we will go over different existing data for event coreference task, both within-document and cross-document. Further data analysis is carried out leading to interesting observations. We concludes this section after literature review and data analysis, which helps give a solid task definition and inspire new ideas tackling the problem of event coreference.

We will follow the definition of terms as described in Chapter 1, which might be different from some papers' term usage. Whenever that happens, we will note it before going into details.

## 2.1 Literature Review

### 2.1.1 Within-document Event Coreference

We will start our survey from Within-document Event Coreference, because most of the work for event coreference currently either focuses on Within-document Event Coreference only, or works for both this and Cross-document Event Coreference. It is also a nature step to do Within-document Event Coreference before the cross-document one.

The first paper addressing the problem of Event Coreference is [32]. This paper focuses on Within-document Event Coreference only. It is a simple extension of their Entity Coreference [25]. The key idea is to describe the world as a hierarchical ontology implemented with the XI knowledge representation language as described in [24], then instantiate a discourse knowledge graph by trying to fit every event mention into this world model frame. The ontology contains entities (referred as "object" in the original paper) with sub-entities, events with sub-events, and attributes associated to each entity or event node. Event Coreference happens as the incremental step during the knowledge graph building process. When each event mention is being translated to predicate-argument representation and added into the knowledge graph, it is compared with a set of events that are already in the graph to decide if it should be either merged to an existing event and update it, or added as a new event. The similarity score is simply derived from the combination of event type distance and event attribute similarities. They only evaluated their system indirectly by switching on and off the event coreference module in their MUC system, showing some improvement. They also noted some good observations. One is that while some events are addressed as single simple events in the text, some events might otherwise be composed by several sub-events. Another observation is that events are referred to in various surface forms, like verb phrases, nominalised forms, or infinitival forms. They could also appear in main clauses, relative clauses, or subordinate clauses.

[22] analyzed event coreference between two sentences in a more linguistic way. They mainly claimed that two sentence could be event coreference only if either these two sentences have a so called ”particularization relation” (one sentence contains additional information compared to the other one) or ”generalization relation” (one sentence does not have additional information but is more general than the other one).

After a couple of years, people started to work on the problem of Within-document Event Coreference again. [17] and [15] described a simple pairwise clustering model to be applied to the task of Within-document Event Coreference, with trained distance function using Maximum Entropy model on ACE2005 data. Features used are focusing on distance, argument, attribute, and other basic features. The attribute mentioned here refers to polarity, modality, genericity, and tense that are annotated in ACE2005 data for events. They claimed that these attributes had been ignored in event coreference community and we should pay more attension to them. They used three metrics to evaluate: MUC F-Measure [45], B-Cubed F-Measure [3], and ECM F-Measure [36]. Their evaluation showed that attribute features helped a lot when using gold event mentions in the Event Coreference task. But they are not effective when using system predicted event mentions.

[16] is another paper they proposed at the same time. They apply the spectral graph clustering algorithm mentioned in [43] into the event mention space. In this space, each event mention is a node, and there are undirected weighted edges representing the confidence of pairs of event mentions being coreferred. Each type forms a graph. They proposed two methods to generate the confidence. One is simply trained maximum entropy model, another one is hand designed confidence function based on several trigger and argument statistics in the training data. Their results showed significant improvement from two very basic baselines (cluster regarding types, or all singletons). Also, the maximum entropy method is a little bit better than the other one.

[5], [7] and [8] are using nonparametric bayesian models to do unsupervised event coreference. It works for both within-document and cross-document coreference. They proposed two nonparametric bayesian models,

one finite feature model and one infinite feature model, to tackle this problem. The experiments were done on ACE2005 and also EventCorefBank(ECB) which was annotated by themselves as described in [9]. They evaluated with B-Cubed F-Measure [3], CEAF F-Measure [36], and pairwise F1. They showed significant improvement from two trivial baselines on both ECB and ACE2005 data. Meanwhile, when comparing to the supervised method described in [17] which is using gold event mentions to get 92.2 percent B-Cubed F-Measure score, they achieved 83.8 percent with no supervision and only system generated event mentions [6].

[11] formalize a method to do event pronoun resolution. [10] was another paper they proposed at the same time using same technique but applied to event noun phrase resolution to events instead. They researched on various lexical and positional features that were used in normal pronoun resolution, and tried them in event pronoun resolution. Syntactic features were also added via tree kernels. SVM was used to do the supervised classification. One trick they used to modify SVM was to use development data to get a best classification threshold instead of 0. Twin-Candidate framwork was applied - namely, a ranking SVM to compare a pair of candidates and choose the one that was preferable. Their experiments showed several things. One was that their SVM threshold tuning trick was effective. Also, twin-cadidate framework was shown to be useful compared to single-candidate framework.

[13] presents a unified model for event coreference. They integrated seven resolvers including both event coreference and object coreference resolvers. The framework was still the typical two-step strategy: mention pair detection and coreference chain formation. SVM with tree kernel and flat kernel was used for mention pair detection, and both best-link and graph-partition clustering algorithms were tried for coreference chain formation. Several tricks were applied in order to improve the performance. One was that object coreference could be used to highly reduce false positive cases for event coreference. A revised training instances selection strategy was shown to be effective.

[37] represents event to be a tuple of time, location, and event description. Then calculate three similarity scores from them, and use a simple dynamically weighting method to weight and generate final similarity score from these three. To be more specific, event description similarity score is given more weights when this score is either really high or really low. Vector space model, tf-idf, and cosine similarity techniques are used for event description similarity score generation. Improvement is shown that the dynamical weighting is better than linear weighting.

[28] proposed a new representation that works for both Within-Document Event Coreference and Cross-Document Event Coreference. It extends Ditributional Semantic Model (DSM) to so called Structured Distributional Semantic Model (SDSM) by generating context word vector for each possible syntactic relation

anchored on the word, which becomes a matrix representation for each word. After the word representation, they proposed a simple method composing words' matrixes to a phrase's matrix. Then, event coreference is simply reduced to comparing the matrix representations of two event mentions. In their experiment, they generated different granularities of representations for event mentions, then varies different distance metrics and score normalization techniques, and finally got 159 different similarity values. Using these feature vectors they trained a final classifier to make the coreference decision. Their method was shown to beat the DSM approach and the SENNA embedding approach [19] on the ECB data and their mannually annotated dataset so called IC Event Coreference Corpus [28]. Notice that they only evaluated using F1 over coreferencing decisions instead of clustering performance. Namely they did not explicitly do the clustering.

[23] is an ongoing work bring up the question of how to handle implicit arguments in Within-document Event Coreference. Another issue they mentioned is event relations - some related events, although not the same type, are decided to be coreferencing. These two problems showed the complexity of event coreference problem, and possibly hinted that this problem is somehow ill-defined. These also proved that event coreference should be considered in a discourse level. They implemented a text understanding system including an event coreference module which implements a set of heuristics. Since it was an ongoing work, they did not show statistics as evaluation results.

[27] addressed Within-document Event Coreference and Cross-document Event Coreference using the same technique. Besides the proposed simple typical event coreference model (rul-based argument extraction, then simple classification on pairs), it pointed out the issue of missing arguments and argument ambiguity. For their task, they annotated their own data from NewsBrief service of the European Media Monitor. Their result proved their claim that contextual feature is important in order to improve the performance.

[14] proposed a new way to generate event mention chains within documents. They formalize the problem of Within-document Event Coreference into two steps. First step is to generate mention pairs. Second step is to generate event mention chains given these pairs. When generating mention pairs, they built several resolvers for different situations. To be more specific, there are objectNP-NP, objectNP-pronoun, eventNP-NP, verb-NP, verb-verb, verb-pronoun, and eventNP-pronoun seven types of resolvers, used in corresponding situations. Then the chain formation step is performed using a modified random walk algorithm. Instead of the termination step, they cared more about the walk path, seeing it as the coreference chain. They claimed that their algorithm could easily incorporate linguistic constraints and preferences. Also, pronoun coreference information and object graph information are incorporated.

[35] proposed a simple supervised pairwise model to do Within-document Event Coreference. The new aspect they added into the procedure is information propagation, trying to do something like joint learning

of event coreference and missing argument completion. Experiment showed some improvement by this information propagation idea when using Pairwise, MUC and BLANC to evaluate, but no improvement, or even drop a bit, when evaluate using B-Cubed and CEAF. Another contribution they made is to point out the fact that event coreference is still not well defined, in the sense that works within event coreference community were using different definitions so that they are not comparable for most of them.

### 2.1.2 Cross-document Event Coreference

Some of the literatures have already been mentioned above. [5] and [7] reduced Cross-document Event Coreference to Within-document Event Coreference by simply creating a meta-document aggregating documents on the same topic, and then run on the meta-document in the way described in last section. [28] naturally works better in Cross-document Event Coreference if compared to Within-document Event Coreference, even though they did not separatedly take care of these two situations. [37], [27] also address the cross-document version in the same way as within-document.

[4] is one of the earliest paper addressing the Cross-document Event Coreference problem. They define Cross-document Event Coreference to be document deduplication. Their approach is simply extending their existing entity coreference system to events across documents. To be specific, they first do Within-document Event Coreference by using their entity coreference system over verbs or nomination forms of the event they are interested about. A coreference chain would be generated for each document. They would extract sentences according to the coreference chains so that they could summarize the chains to Vector Space Model representations in order to compare across documents, and finally generate cross-document coreference chains. In order to utilize the existing evaluation algorithms for Within-document Event Coreference, which is MUC, they used the trick of generating a meta-document by concatenating all the documents in a same topic cluster, and took the coreference chains we got and evaluate as if dealing with within-document situation. They used their own data set. Another observation they made is the interannotator agreement on annotating their data is 95 percent F score, which showed that cross-document coreference as document deduplication is a doable task.

[44] proposed a simple method to do cross-doc event coreference on plot summaries. The main idea was to match events' attribute instead of event words themselves. They showed simple preliminary results.

[34] was focusing on the task of cross-docuement event fusion, which has cross-document event coreference as an important sub-process. Their method was also applying the popular two-step framework - mention pair similarity identification to mention pair clustering. They added a third phase, event fusion, after the clustering step. They used flat features and structure features to do mention pair identification with tree

kernel. Hierarchical clustering algorithms were further applied.

[33] proposed a method learning Cross-document Event Coreference and Entity Coreference jointly. The main idea is to put events and entities in the same space and iteratively cluster and regenerate features from them. The regeneration of features based on current clustering allows the information to flow between entities and events so that they would improve each other. For example, some semantic role feature based on newer event clustering may result in different role feature for some other entities leading to better entity clustering. This kind of feedback should also work in the opposite direction. Note that they chose not to treat entity and event separately, but seperates verbal and nominal expressions. They were doing careful babystep clustering in each iteration, only merging the most possible pair of clusters. The algorithm ends when no merging brings benefits. They score each merge using a trained linear regressor, which is trained iteratively with algorithmatically selected training data and then linearly interpolated. In order to try to avoid local maximum as all hill-climbing algorithms do, they initialized their training model with hints from the deterministic sieves. A trick in their algorithm is that they first used nonparametric methods to do document clustering, and then appied their high precision deterministic sieves for entity coreference to reduce the search space. To evaluate their algorithm effectively, they extended ECB dataset, and used different metrics to evaluate, including MUC, B-Cubed, CEAF, BLANC [41], and CoNLL F1 [39]. The results showed significant improvement over strong entity coreference and event coreference baselines.

[20] implements the idea of comparing time, location and participants of events with soft matching. Here, by soft matching, they refered to different partial coreference in location, time range, participants. To be more specific, they mentioned the scriptal relation, is-a relation, and membership relation. They deliberately chose not to use machine learning techniques because they claimed to identify the contribution of the soft matching. Instead, they used simple similarity measure to score the partial matching using different available NLP tools like WordNet and Lemmatizer etc. The experiment was carried out on ECB dataset. Cross-document Event Coreference was performed on previously clustered topic document sets. They showed slight improvement in precision over a simple lemma-based baseline.

[2] More interesting than the algorithm they proposed, the explicitly defined three-layer event ontology clearly helped disambiguate some terms that have been used inside the community of event coreference. The three layers are Formalism, Realization, and Text. In our own words, we could see them as event types, event instances, and event mentions. Formalism layer (event types) defines the general frame of each event type. Relations among events, like sub-event, should be addressed in this layer. Realization layer (event instances) represents concrete events with the event frame defined in Formalism layer filled. Event coreference should happen on this layer, comparing similarity of arguments. There are one-to-many links from Formalism layer

to Realization layer. Text layer (event mentions) is original text span mentioning events. Many-to-many links between Realization layer and Text layer exist, due to the fact that sometimes a mention could mention two events or more. Note that people also address clustering of event mentions into event instances as event coreference. The algorithm they proposed is standard: partition corpus first (instead of topic, they partition according to time window), event mention identification, event type classification (using WordNet), then do incremental clustering of mentions (compare mention against existing clusters based on participants). They annotated AQUAINT TimeML corpus.

### 2.1.3 Other Event Coreference Paper

[29] is a paper deeply analyzing the NP4E corpus for event coreference. They made two important observations. One is that 70 percent of coreferencing events' corresponding argument slots are filled with entities that have indirect anaphoric relationship (e.g. part-of, set-member, subset, etc.) which is not a coreference relationship. Another is that lots of events do not have all of their argument slots filled.

[26] describes an annotation system they designed in order to annotate both entities and events for cross coreference at the same time. This system asks annotators to create an event template for a certain event, and then put all the related event mentions under it and also try to fill the template. They allowed entities and events to be linked to Wikipedia or DBpedia.

[30] claimed that event coreference is more complicated than the way we used to treat them. The proposed the idea of quasi-identity so that we could further define full coreference, partial coreference and non coreference. Two mentions are full coreferring when they have lexical identity, or are synonym, or one is synonym of wide-reading of the other one, or are paraphrase relation or one is another's pronoun. Two mentions are partial coreferring when they have member-of relation, or sub-event relation, or part-of relation. Moreover, they claimed that communication events should be treated differently since there could be inconsistent information. Their contribution is the two datasets further annotated under this proposal - Intelligence Community Corpus (IC) and the Biography Corpus (Bio).

[1] proposes an idea how to construct subevent relations among within-document event mentions. which is expected to help improve Within-document Event Coreference since it put aside those mentions related to each other as subevents instead of full event coreference. Their proposed method contains two stages. The first stage is a logistic regression model classifying mention pairs into four types of links: full coreference, subevent parent-child, subevent sister, or no coreference. Then a voting strategy is used to finally construct the subevent relations based onn subevent sister relations we got from stage 1 (because this relation seems to be much more precise compared to other relations as they observed in experiments).

[31] proposes evaluation methods for partial event coreference - here including sub-event relation and membership relation. Besides the evaluation method itself, they noted the importance of distinguishing full coreference and partial coreference, which is pretty interesting.

[21] released the ECB+ corpus, an extension of the ECB corpus that has been broadly used in event coreference community.

## 2.2 Conclusions and Observations

From the survey we can see that, all of the proposed solutions from related publications, for both within-doc and cross-doc situations, decompose this problem to two parts: the similarity metric and the clustering algorithm.

For the clustering problem, most of the solutions choose to use incremental pairwise clustering on event mentions. Besides these solutions, [16], [13] choose to use spectral graph clustering algorithm. [5], [7] and [8] use non-parametric bayesian model to do the clustering. [33] uses linear regressor to do the merging. [34] tries to use hierarchical clustering to solve the problem.

The distance measures in different clustering algorithms are typically carefully trained similarity metrics. Besides that, [28], [4] and [37] use euclidean distance and cosine distance in its clustering algorithm instead of trained similarity metrics.

Besides the details in algorithms, we can also learn about the typical experiment settings. To be more specific, experiment data used are usually ACE2005 and ECB. There are also related works using OntoNote, IC Event Coreference Corpus, Extended ECB, and AQUAINT TimeML, where some of them are annotated by researchers themselves. The most popular evaluation methods used to score the clustering algorithms' performance are MUC, B3, CEAF and BLANC.

# Chapter 3

# The Event Coreference Problem

## 3.1 Event Coreference and Typical Solution

The Event Coreference problem is the task to cluster all event mentions discussing the same event. Each event mention contains at least one event trigger, typically a verb or its nominalization, and some of the event arguments. An event usually have these event arguments: event time, event location, event subject, and some event objects depending on different types of events. An event mention is not guaranteed to contain all the event arguments. Actually, event arguments are rather likely to be spread to different event mentions throughout an article. Therefore the Within-Doc Event Coreference aims to aggregate these event mentions so the event arguments information about the same event could be put together. The Cross-Doc Event Coreference is a similar task, where the only difference is that Cross-Doc Event Coreference operates on event mentions across different documents.

Typical ways to solve this problem mostly involve two crucial components. One is the similarity metric between event mentions. Normally it is carefully designed and trained to give similarity scores to event mention pairs. Another is the clustering algorithm using the similarity metric as the distance, which tries to maximize the overall similarity.

## 3.2 Problem of Typical Solution

There might be problems in both the similarity metric and clustering algorithm parts of the typical formalization.

### 3.2.1 Coreference Does Not Imply Mention Similarity

As typical coreference models are formalized to optimize the total links similarity score, we noticed that this might not quite apply in the event coreference case.

The key observation is that two event mentions that do not look similar to each other might actually be

| ID | Example |
|---|---|
| *em1* | Word comes from People magazine and other celebrity news outlets that Tara Reid has **entered** the Promises Treatment Center in Malibu. |
| *em2* | The original trainwreck Tara Reid's publicist confirmed that the actress Reid was **admitted** into Promises Treatment Center in Los Angeles, California. |
| *em3* | Reid **checked** herself in on Tuesday, December 9, 2008. |
| *em4* | In opening statements Tuesday, public defender Peter Lynch **said** then-15-year-old Timmons was at the scene of the shooting and had a gun. |
| *em5* | The mother of a 10-year-old girl killed by a stray bullet on an Albany street last year **said** she heard a loud sound before her daughter cried. |

Table 3.1: Examples from ECB

coreferent. As we have mentioned before, an event contains several components: event time, event location, event subject, event object, and other optional event participants. But there is no guarantee that one event mention will contain all of these informaiton. In fact, it is common that one event mention contains part of the event components, while the other mentions containing the rest. In this case, these mentions will not look very similar to each other, but they should still be clustered together since they have different pieces of information and could be put together into a single event containing complete information.

For example in Table 3.1, the two bolded event mentions in *em2* and *em3* are actually coreferent to each other. However, it is not likely to be scored highly by most trained similarity metrics, since *em2* contains an event object information that *em3* does not have, while *em3* contains the event time information which *em2* does not have.

It might not be easy to train a mention-based similarity metric to capture these kind of cases, which which occurs frequently in texts due to people's tendency to convey complicated information through out multiple sentences while avoiding duplicated information. This suggests that we should use a deeper representation and a more complicated metric that goes further from the simple similarity metric.

### 3.2.2 Linking Order Does Matter

Ideally, most of the clustering algorithms used in the typical formalization should think about the balance between accuracy and efficiency. On the one hand, to guarantee efficiency, it might seem acceptable to do $O(n^2)$ comparisons between every pair of event mention once. Some might even do less comparisons to reach similar accuracy.

However, the accuracy of certain cases might not be guaranteed. For example, if three coreferent mentions separately contain information "ABD" "BCD" "CA". The similarity metric is "if contains two or more common letters then coreferent". Since we only do every comparison once, if we try to link "CA" with any other mentions, then “CA” will never be merged with other mentions.

One solution is to re-order the comparisons. Link “ABD” and “BCD” before “CA” will solve this problem without extra run time. This is also the solution we will go for in our proposed algorithm. The key is to use event mentions' semantic level information to guide the linking order.

Most clustering algorithms applied to the event coreference problem does not use semantic level information contained within event mentions to guide the order they link, or not even care about the order at all. For those who ignore the order at all, the false assumption is that all data points to be clustered are independent, which – in our case – is far from true.

## 3.3 Proposed Solution

To solve the problems mentioned above, We propose a similarity metric that depends more on event arguments, and two clustering algorithms based on the idea of ranking the clusters with the amount of information they convey. The Informative-First Clustering algorithm (IFC) clearly explains this idea, and is used in within-document event coreference tasks. The Topic-Side Event Clustering algorithm (TSEC) is an extension of IFC, meaning the main idea is still appliable in cross-document event coreference cases. The amount of information conveyed by an event mention cluster is denoted as “Event Information”. which we will see how to estimate in the next chapter.

The main idea is to add another step to the typical solution: aside from a similarity metric and the clustering algorithm itself, we also need a metric to evaluate the amount of information conveyed in an event mention cluster. Based on this value, our new clustering algorithm will be guided so that it always compares the pair of clusters that have the highest information amount first.

In the following chapters, we will separately discuss about these components. In Chapter 4, we will discuss how to calculate the event information value of a cluster. In Chapter 5, we will talk about the similarity metric – it should be a bit different than commonly used similarity metrics due to nature of our algorithms. In Chapter 6 and 7, IFC and TSEC will be discussed. In Chapter 8 we will show a running system (EVCO) using these algorithms.

# Chapter 4

# Event Information

## 4.1 Overview

Event Information is the key idea introduced in this thesis. All of the algorithms proposed use it to decide clusters merging order and also the similarity threshold.

Each cluster should have an Event Information score, indicating whether the information about an event conveyed by this cluster is sufficient. To be specific, a cluster having all the components of the event where all of the components have high trustworthiness should be given a high Event Information score. Namely, we value the completness and trustworthiness of the event information contained within a cluster.

## 4.2 Representation

Figure 4.1 shows how we represent an event, or a cluster that we use in our algorithms. Basically, an event has a trigger, which connects all the sufficient and neccessary event components identifying an event - event time, event location, event subject, and possibly two event objects. For each component, we keep a list of candidate mentions with scores indicating the trustworthiness of corresponding event component mention. The list of candidate mentions are sorted by the scores, and the one with highest score is treated as a representative of the event component.

Note that a single event mention could be treated as a singleton cluster. Whenever we see an event mention, we identify the trigger, then extract event components from this event mention, with confidence scores. After that, we have enough knowledge to create the representation we described above, with each component's candidate list having only one or zero candidate.

Then, when we want to merge two clusters in this representation (we will make sure that it only happens when there are no conflicts in event trigger and event components), we simply merge corresponding component list while maintaining the sorted order.

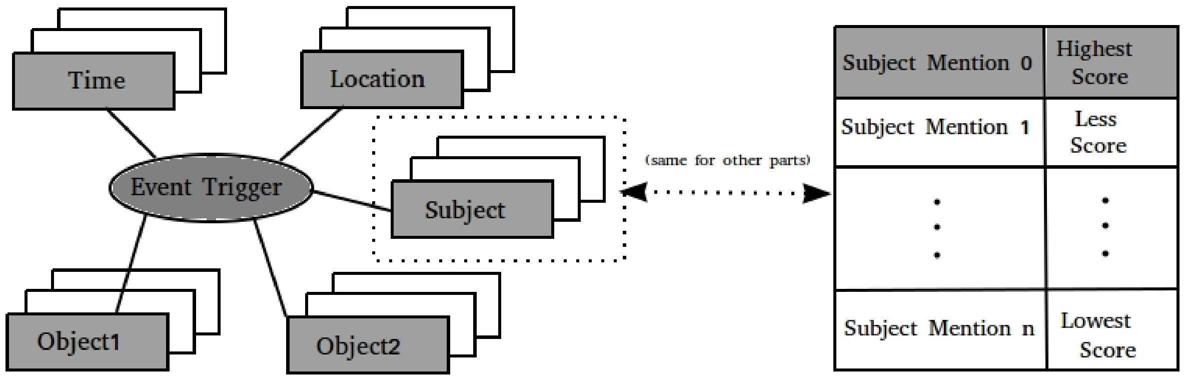

Figure 4.1: Event Representation

## 4.3 Calculation

Based on this proposed representation, we can invent a reasonable metric to calculate the Event Information score for an event cluster. As shown in equation 4.1 below, the Event Information is the weighted sum of all event components' trustworthiness. To be more specific, $\Phi$ is the set of event component, $w_i$ is the information weight for the corresponding event component $i$. Lastly, $T(e_i)$ represents the trustworthiness of component $i$ in event cluster $e$.

$$I(e) = \sum_{i \in \Phi} w_i T(e_i) \tag{4.1}$$

where $\forall i, T(e_i) \in [0,1], w_i \in [0,1]; \sum_i w_i = 1$; $\Phi = \{time, location, subject, object1, object2\}$

The trustworthiness could be calculated as described in equation 4.2. The figure 4.2 gives a graphical explanation of the component trustworthiness calculation.

$$T(l) = \begin{cases} 0 & [n = 0] \\ l[0] & [n = 1] \\ T(l[0, n-2]) + l[n-1](1 - T(l[0, n-2])) & [n \geq 2] \end{cases} \tag{4.2}$$

where $n$ is the length of the input list $l$.

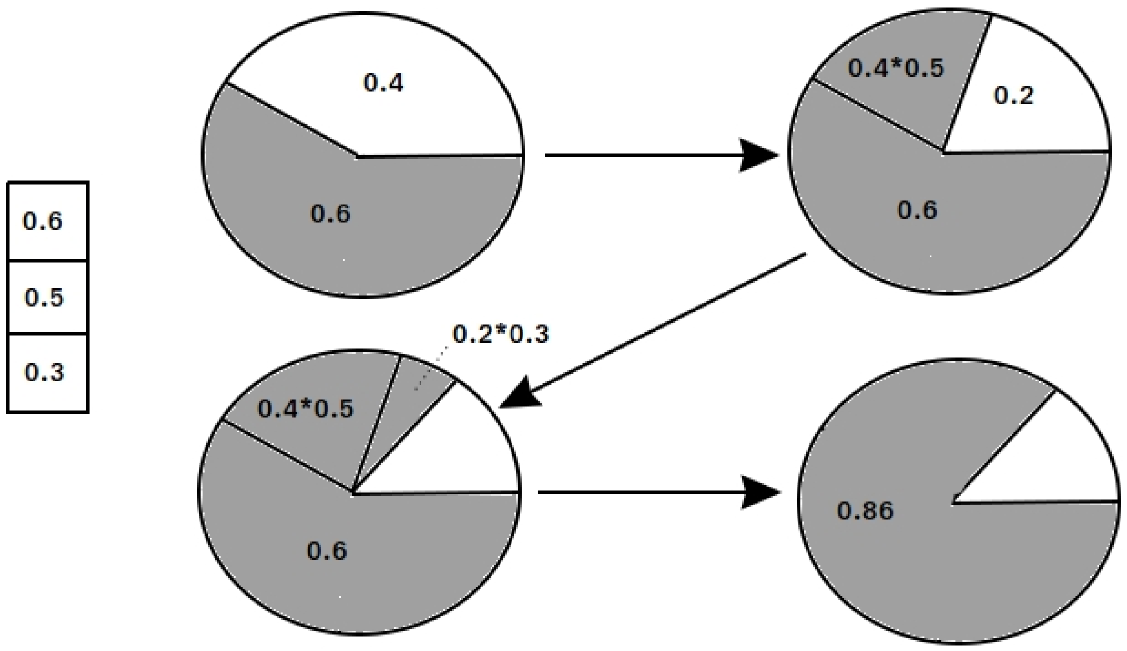

Figure 4.2: Component trustworthiness calculation example

As one may notice from the equation, an event cluster that contains all event components where each event component has high trustworthiness will finally receive a high event information score. Those who miss some necessary event components or has low components trustworthiness will finally receive a relatively low event information score. The weights for different event components could be dfferent based on different event types, which could be specified by user or some frames.

The score for each candidate event component mention could be the confidence of predicting such mention is the corresponding component of the event mention.

## 4.4 Event Arguments Extraction and Normalization

As one may see from last chapter, event arguments are important in order to solve this problem correctly. Specifically, these five arguments are crucial: time, location, subject, object1, and object2. We will do similar things when building extractors for each of these five arguments.

To obtain all of the necessary event arguments, we use the Illinois SRL tool to get Semantic Role Labeling annotation on the event mention sentence. Then we align the event trigger with the predicates identified by the SRL annotator. After the predicate alignment, we can align these arguments: "AM-TMP" as event time, "AM-LOC" as event location, "A0" as event subject, "A1" as event object1, and "A2" as event object2.

Then we need to get trustworthiness scores for all the extracted event arguments. The confidence score of predicting each event argument could satisfy this role. However, due to the fact that illinois SRL tool makes a final decision by global inference so it will not provide a meaningful confidence score, we will train classifiers to get the confidence score instead. Specifically, we will train binary classifiers using ACE2005's event mention argument information. These binary classifiers is able to decide how likely an event argument mention is to be the event time/location/subject/object1/object2. Then, the confidence score generated with these classifiers will be used as the trustworthiness score of an event argument mention.

Lastly, it is important to normalize these event argument in order to the further comparison when generating the similarity will be more accurate. The illinois time tool would be used to normalize the time arguments. Unfortunately, for the data we are experimenting on, the ECB data, does not have document creation date for reference. We also use illinois entity coreference tool to resolve pronouns and other entity mentions by preprocess the documents.

# Chapter 5

# Core Similarity

## 5.1 Overview

In this chapter, we will discuss about an new idea of the similarity metric, which takes into consideration the missing event arguments in an event mention.

Although an event is uniquely identified with the trigger along with the spatiotemporal arguments and participant arguments, it is likely that a single event mention will not contain all of these information at once, especially for the within-doc case, due to the nature how people write articles. Actually, this is also why we want to do event coreference - we want to aggregate information about the same event from different event mentions.

For example, in table 5.1, we can see that the first event mention em1 has "fired" as its trigger, with subject "Timothy", object1 "a shot", and object2 "them". Meanwhile, the second event mention em2 has the location information, but no information about who Tim fired at. Both mentions are missing some of the necessary event arguments to identify an event.

There are even more extreme cases where two mentions of a same event have no overlapping event arguments, but linked together only with the trigger. For example, em3 and em4 are annotated as coreferent in ECB data set. However, they only have overlaps in their triggers. Specifically, em3 has the subject and object1 of the event, while em4 only contains the object2 argument. In cases like this, the two sentences might look totally different so that even a well designed sentence-level similarity metric will assign low score

| **ID** | **Example** |
|---|---|
| *em1* | Timothy **fired** a shot at them. |
| *em2* | Tim **fired** the deadly shot from an intersection down the street. |
| *em3* | Hewlett-Packard is rolling the dice on a 13.2 billion **acquisition** of technology services provider Electronic Data Systems. |
| *em4* | The **deal**, when completed, will give it about 7 percent of the technology services market. |

Table 5.1: Examples from ECB

to this pair. However, if the similarity metric could notice the fact that, one main reason that these two sentences look so different is because they do not have any event arguments overlapped, it could punish the pair of mentions less thus generates a reasonable score.

This kind of situations could be taken care of with our proposed similarity metric. Our clustering algorithms discussed in following chapters also try to further handle these cases well.

## 5.2 Proposed Solution

### 5.2.1 Scoring Function

When we try to merge two clusters in the representation described in previous chapters, we will need to use the similarity metric described here. The goal is to ensure if we decide to merge two clusters, their overlapping event arguments agree with each other to certain degree. To be specific, we will need to compare the event trigger, event time, event location, event subject, and the event object1 and event object2 separately.

$$S(\mathbf{C}_1, \mathbf{C}_2) = (1 - w_{arg}) S(\mathbf{C}_1.mention, \mathbf{C}_2.mention) + w_{arg} \min_{i \in \Phi_1 \cap \Phi_2} S(\mathbf{C}_1.i, \mathbf{C}_2.i) \tag{5.1}$$

where $\Phi_k$ represents $\mathbf{C}_k$'s valid event arguments set. $\forall k, \Phi_k \subset \{time, location, subject, object1, object2\}$

The equation 5.1 describes the similarity metric that we propose. The main difference it has when compared to other learned similarity metric is that it not only check the similarity between event mentions, but also check the similarity between all the overlapping event arguments of these two event mentions, and include these similarity results into the overall similarity score as a regularization. Specifically, we see the lowest argument similarity score as the “argument agreement score”. If the “argument agreement score” is high, it means all of the overlapping event arguments have high similarity. On the other hand, if the “argument agreement score” is low, it means that at least one of the overlapping event argument pairs has low similarity, making these two event mentions less likely to be from the same event mention cluster. Notice that we ignore those event arguments that only exist in one or none of the event mention clusters being compared. This is due to the nature of missing event arguments we discussed about in previous chapters. When one event mention cluster is missing a certain event argument, this event argument should not contribute to the “argument agreement score”, since it brings no conflicts.

| Feature | Explanation |
|---|---|
| ExactMatch | If the two mentions are exactly the same, set to 1. Otherwise 0. |
| SubStringMatch | If the two mentions contain overlapping words, set to 1. Otherwise 0. |
| LemmaMatch | If the two mentions' lemma matches, set to 1. Otherwise 0. |
| SubStringLemmaMatch | If the two mentions contain overlapping lemmatized words, set to 1. Otherwise 0. |
| NESim | The similarity score of these mentions in NESim. |
| WNSim | The similarity score of these mentions in WNSim. |
| LLM | The similarity score generated by LLM similarity. |
| InSameSentence | If two mentions are from the same sentence, set to 1. Otherwise 0. |

Table 5.2: Preliminary features used to train event component pair similarity classifiers

### 5.2.2 Preprocessing and Feature Engineering

Before passing the event mention cluster pairs into this similarity scorer, there are also some preprocessing need to be done. First of all, as is mentioned before, the event arguments are extracted using SRL - we run SRL on the sentence that contains the annotated event mention trigger, then try to align the event mention trigger with one of the predicates of SRL. After aligning it, we treat the SRL-annotated arguments of the predicate as our extracted arguments. Specifically, AM-TMP is the time argument, AM-LOC is the location argument, AM0 is the subject, AM1 is the object1 and AM2 is the object2. We use illinois SRL annotation tool [40] for this job.

Another preprocessing we do is to run illinois entity coreference tool [12] [38] over the input documents. After SRL alignment, we try to align our extracted entity arguments, including location, subject, object1 and object2, to the entity coreference results in order to resolve a normalized entity mention as our event argument so that our similarity metric could score more accurately. This step is important because people always try to use different mentions to refer to the same entity when they write an article. Variations include pronouns, paraphrases, abbreviations etc. We also observe a lot of such cases appearing in our ECB data set. Since our similarity metrics look at event argument mentions directly instead of the full document, these variations will be difficult instances for our similarity scorer to understand. Resolving the entity coreference as a preprocessing step is the most reasonable solution that tackles all these problem at once. We see this as a normalization step.

Another preprocessing that should be done is time normalization. Basically, what this means is to resolve relative time mentions (e.g. tomorrow, last year, ten days ago, etc.) to a normalized date or time (e.g. 2015/12/08). With these normalized time mentions, we could compare and score them much more easily. Illinois time tool woould be a perfect fit for this task [46]. Unfortunately, there is no document creation date for our ECB data set, so that there is no reference date that we could use.

We use ACE2005 to train all the trigger and argument similarity classifiers used above. Specifically, we will train the trigger similarity classifier, time similarity classifier for time argument comparison, location similarity classifier for location argument comparison, and entity similarity classifier for subject and other objects. The LBJava tool [42] from UIUC is used to train all of these similarity classifiers. Different pairwise features are used to train these classifiers as shown in table 5.2. These are basically syntactic and semantic level features. All of the required annotations are generated via UIUC's Curator [18]. Currently we only have this small set of features for experiment use. Better feature engineering should be carried out in the future.

# Chapter 6

# Within-Doc Event Coreference: Informative-First Clustering (IFC)

## 6.1 Overview

We choose to still follow the typical formalization; But we also propose a new type of clustering algorithms - the Informative-First Clustering (IFC), to solve the problems presented previously. We will start with within-document event coreference task when describing the algorithm. The cross-document event coreference version of this algorthm is described in the following TSEC algorithm chapter.

**Algorithm 1** Informative-First Clustering (IFC)

```
**Input:** $\mathbf{M}_D$ (the set of detected event mentions within document $D$)
**Output:** $\mathbf{\Gamma}_D$ (the final clustering of event mentions in $D$)
$\mathbf{\Gamma}_D = \varnothing$
**for** $\forall \mathbf{m} \in \mathbf{M}_D$ **do**
    $\mathbf{C} = \{\mathbf{m}\}$
    $\mathbf{C}.info = calInfoScore(\mathbf{C})$
    $\mathbf{\Gamma}_D.add(\mathbf{C})$
**end for**
$\mathbf{\Gamma}_D.sortByInfo()$                    $\triangleright$ // Clustering initialized with sorted singleton clusters.
**while** $!\mathbf{\Gamma}_D.finalized()$ **do**
    $\mathbf{C}_1, \mathbf{C}_2 = \mathbf{\Gamma}_D.getNextUntouchedMaxInfoPair()$
    **if** $calCoreSimilarity(\mathbf{C}_1, \mathbf{C}_2) \geq \theta(\mathbf{C}_1, \mathbf{C}_2)$ **then**
        $\mathbf{C} = merge(\mathbf{C}_1, \mathbf{C}_2)$
        $\mathbf{C}.info = calInfoScore(\mathbf{C})$
        $\mathbf{\Gamma}_D.insertByInfo(\mathbf{C})$
        $\mathbf{\Gamma}_D.remove(\mathbf{C}_1)$
        $\mathbf{\Gamma}_D.remove(\mathbf{C}_2)$
    **end if**
**end while**                    $\triangleright$ // Loop until every cluster pairs compared.
```

In general, the Algorithm 1 describes IFC for within-document event coreference. We will go into details in the following sections. Basically, what this algorithm does is to first initialize the clustering with a list of singleton clusters sorted by the calculated event information score, in which each cluster contains one detected event mention within this document. Then we will start with the top two clusters (because they have the highest information sum), compare their core similarity (if the trigger and shared event arguments align with each other then they are similar). Once we decide to merge them, we will take out the original clusters and merge into one large cluster, calculate the event information score for the new cluster, then put it back to the clustering respecting the sorted order. We will find the next pair of clusters that have the highest event information sum and are not yet compared, apply the same operations on this pair. Keep on doing this until no cluster pair available for comparison. The clustering will be the final output.

The section below will cover the details in how to do the clustering efficiently provided with the calculated event information score of every cluster. Specifically, the $finalized()$ function, the $getNextUntouched$-$MaxInfoPair()$ function, the $merge()$, the $sortByInfo()$ and the $insertByInfo()$ functions will be explained.

## 6.2 Algorithm Details

To ensure this algorithm runs efficiently, we need to pay attention to two things: the sorting and the updating. The final time complexity of IFC is $O(n^4 logn)$.

### 6.2.1 Sorting

The X+Y Sorting algorithm will be used consistently throughout this thesis.

After initializing the singleton clusters in order (with the function $sortByInfo()$ which determines that it keeps the order of event information), what we are really doing is an X+Y sorting problem. Namely, we sort all possible pairs of clusters according to their event information sum.

The X+Y sorting problem currently remains an open problem. It is agreed that current best algorithm reaches the run time of $O(n^2 logn)$. Actually, just by running merge sort or quick sort over alll possible cluster pairs could acheive this time complexity. Moreover, with a clever design, the space complexity could be constrainted to $O(n)$. Since merge sort would require $O(n^4)$ space and quick sort would have the worst run time of $O(n^4)$, we would like to use the cleverer algorithm which always runs in $O(n^2 logn)$ time and $O(n)$ space.

Figure 6.1 shows an example run of the algorithm we use for this sorting problem. The X+Y sorting problem is to traverse the universe of pairs using the sum of numbers as a guide, going from high sum to low sum. We keep a priority queue to keep track of the max sum next cell. In the figure, this means that whenever we mark a cell black, we will enqueue the gray cells into the priority queue (the one below the black cell, and the one on its right, if they are still white cells). We start from the highest sum pair - in the example, it is the cell in the up left corner, and enqueue it. Then we keep dequeueing and mark the

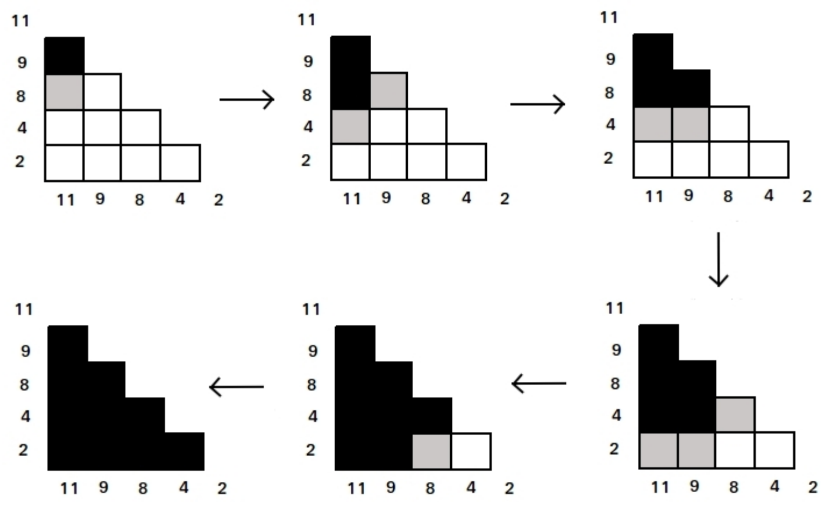

Figure 6.1: Best algorithm currently for X+Y sorting algorithm.

popped cell black (which possibly triggers more enqueues of gray cells). The algorithm finalizes when there are nothing to dequeue.

This algorithm goes through all $O(n^2)$ cells once. Each time it will do constant times of $O(logn)$ enqueues and dequeues. Therefore, the time complexity is $O(n^2logn)$. The space complexity is bounded by the max size of the priority queue. Since the priority queue always keeps the cell that are on the boundary between marked and unmarked cells, and the boundary length is $O(n)$, the space complexity is thus $O(n)$.

This sorting algorithm is applied in the IFC algorithm described previously. The $finalized()$ function returns true if there is nothing to dequeue in the priority queue. The $getNextUntouchedMaxInfoPair()$ is simply dequeue the priority queue which returns the current max sum cell on the boundary.

### 6.2.2 Updating

Each time we choose to $merge()$ two clusters, we need to update the clustering accordingly. Basically these two clusters should be merged and a new cluster will be generated. After calculating the event information of the new cluster, we will drop the two original clusters and put the new cluster into the clustering. The order should still be kept. Then we will run the X+Y sorting algorithm mentioned above on the new clustering from the beginning cell until the next merge. In order to skip those pairs that we have already processed, we will cache the results and always check if cache hit or not before actually calculating the similarity.

The maximum number of merges is $O(n^2)$. Namely, we might run the $O(n^2logn)$ algorithm for $O(n^2)$ times. Thus, in total it will take $n^4logn$ time for the whole IFC algorithm.

### 6.2.3 Threshold

For the threshold we use to decide if two events are similar or not, we can base it on the total event information value. The less information they contain, the higher similarity score they should achieve in order to pass the decision boundary. This idea will also be applied in TSEC algorithm.

# Chapter 7

# Cross-Doc Event Coreference: Topic-Side Event Clustering (TSEC)

## 7.1 Overview

The IFC algorithm introduced in the previous chapter fits well for the within-document event coreference task due to the nature of it – number of event mentions contained within one document is positively related to length of document, which typically will not be too long. Therefore, the $O(n^4 logn)$ algorithm will still run in reasonable time. However, for the cross-document event coreference case, there might not be a limitation on the number of documents being processed at one run. Thus, a linearly scalable algorithm for this task is necessary. That is why we introduce the following algorithm.

The Topic-Side Event Clustering algorithm (TSEC) is an extention of the IFC algorithm. The main difference is that it is based on the assumption that each document contains one main topic event. The document might also have mentions talking about other related events, but the total amount of information conveyed is significantly less than the main topic event. Also, we assume that clusters from the same document will not be coreferent to each other. Implementing these heuristics, we can have a good approximation of IFC with a good run time and accuracy performance.

Throughout this thesis, we will denote the topic event of a document to be the event mention cluster that has the highest event information score. All other event mention clusters within this document are called side events.

## 7.2 Algorithm

The TSEC algorithm could be split into three steps. First, we will try to compare all topic events in the same order as in IFC. When a pair of clusters coreferent each other, we will somehow merge these two documents. When no more topic events could be coreferent, we move on to the second step. The second step is to try connect every topic event to any side events that are not from the same document. The order is still the same as in IFC. Lastly, we will attempt to link every side event pairs that are not from the same document.

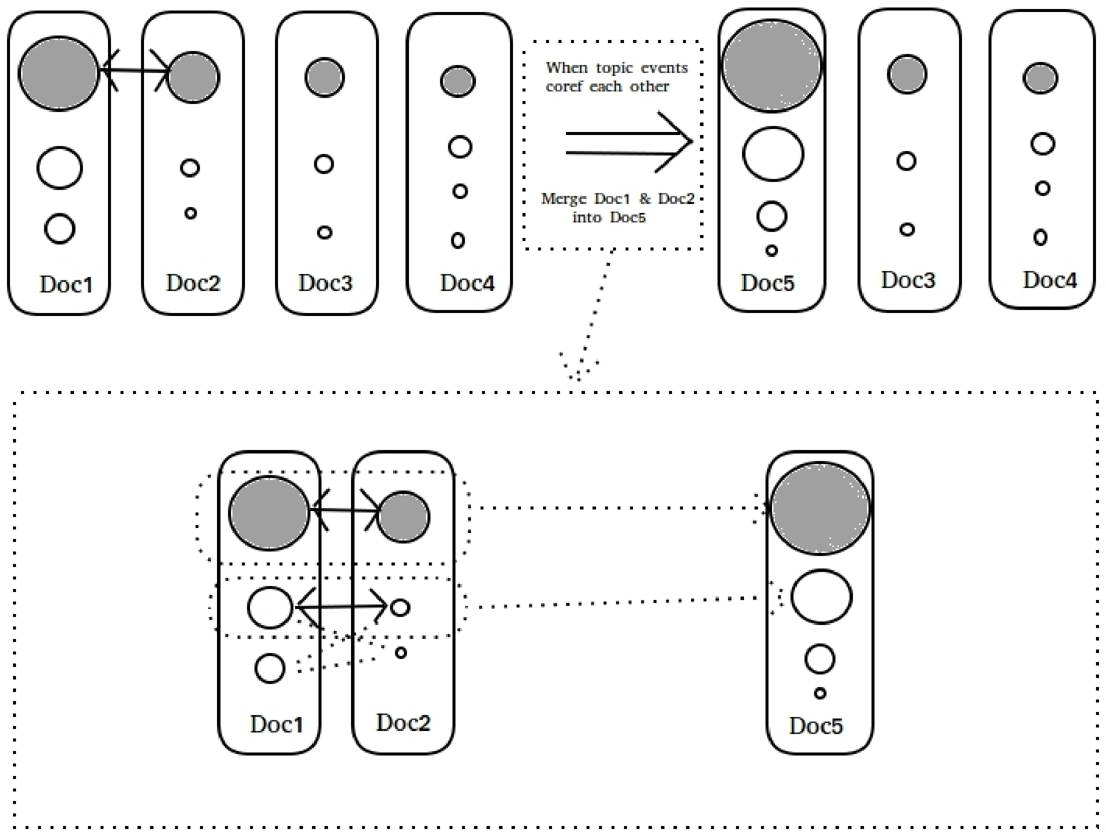
Doc1
Doc2
Doc3
Doc4
When topic events coref each other
Merge Doc1 & Doc2 into Doc5
Doc5
Doc3
Doc4
Doc1
Doc2
Doc5

Figure 7.1: TSEC step 1: Topic Event and Topic Event

This is almost the same as IFC. We will explain the algorithm in detail below.

### 7.2.1 Topic Event and Topic Event

The first step in TSEC is attempt to make links among topic events. This is because topic events contained within documents contain the most event information. Thus, trying to link them first will bring us more reliability.

The input to TSEC is the list of documents with all the within-document event mentions being clustered using IFC. The first step starts in an IFC-like manner, which orders the topic events according to their event information amount. Then choose pairs of topic events to compare in the same way as in IFC. When the core similarity metric decides that two topic events should be coreferent, the two corresponding documents will be merged. Specifically, the two topic events will be merged into one, then the side events in two documents will be compared in informative-first order, and merged when necessary. This step makes sense

**Algorithm 2** Topic-Side Event Clustering (TSEC) - Step 1: Topic & Topic

```
**Input:** $\mathbf{\Gamma}_0 = \{\mathbf{\Gamma}_D\}_{D\in\mathbf{\Delta}}$ (IFC-generated clusterings for all documents in data $\mathbf{\Delta}$)
**Output:** $\mathbf{\Gamma}_1$ (clusterings for merged documents)
$\mathbf{\Gamma}_1 = \mathbf{\Gamma}_0.sortByTopicInfo()$
while $!\mathbf{\Gamma}_1.finalized()$ do
    $\mathbf{\Gamma}_{D_1}, \mathbf{\Gamma}_{D_2} = \mathbf{\Gamma}_1.getNextUntouchedMaxTopicInfoDocPair()$
    if $calCoreSimilarity(\mathbf{\Gamma}_{D_1}.\mathbf{C}_{topic}, \mathbf{\Gamma}_{D_2}.\mathbf{C}_{topic}) \geq \theta(\mathbf{\Gamma}_{D_1}.\mathbf{C}_{topic}, \mathbf{\Gamma}_{D_2}.\mathbf{C}_{topic})$ then
        $\mathbf{\Gamma}_{D_{merged}}.\mathbf{C}_{topic} = merge(\mathbf{\Gamma}_{D_1}.\mathbf{C}_{topic}, \mathbf{\Gamma}_{D_2}.\mathbf{C}_{topic})$
        $\mathbf{\Gamma}_{D_{merged}}.\mathbf{C}_{topic}.info = calInfoScore(\mathbf{\Gamma}_{D_{merged}}.\mathbf{C}_{topic})$
        $\mathbf{\Gamma}_{D_{merged}}.mergeSideEvents(\mathbf{\Gamma}_{D_1}, \mathbf{\Gamma}_{D_2})$        ▷ //X+Y sorting, 1-1 bipartite match
        $\mathbf{\Gamma}_1.insertByTopicInfo(\mathbf{\Gamma}_{D_{merged}})$
        $\mathbf{\Gamma}_1.remove(\mathbf{\Gamma}_{D_1})$
        $\mathbf{\Gamma}_1.remove(\mathbf{\Gamma}_{D_2})$
    end if
end while
```

because two documents talking about the same topic event are likely to talk about similar side events too. When can choose to allow or forbid side events from the same document to link each other, depending on the trade off between how much we trust our within-document coreference results and how much efficiency we want to gain - if we trust the within-document event coreference result generated with IFC, then we can ban the side events from same document from linking each other, so that it becomes a one-to-one bipartite graph. Otherwise, if we allow side events from the same document to be merged in this step, we are actually allowing cross-document event coreference information to help improve within-document event coreference result, which could be interesting. In a graphical view, this instead generates a many-to-many bipartite graph.

The idea of document merge is shown in Figure 7.1. Each rectangle represents a document. Each circle inside a document represents an event mention cluster. The dark one is the topic event for the document - namely, the cluster that has the highest amount of event information.

The pseudocode is explained in Algorithm 2. You may notice that it looks like IFC algorithm a lot. We use X+Y sorting to decide the order of choosing document pairs, where X and Y are both the topic list. The *mergeSideEvents*() function needs some more explanation: Basically, we still want to try to link pair of side event clusters from different document in informative-first order, which could be achieved by using another X+Y sorting where we set one document's side event list to be X and the other's to be Y. Because we want to maintain an one-to-one bipartite matching so that no side events from the same document will be merged, whenever we choose to merge a pair of side events, we will take them out so that they will not be merged to any other side events in this step.

### 7.2.2 Topic Event and Side Event

The second step in TSEC is attempt to link side events to topic events. For each topic event, we will compare it to all side events that are not in the same document. We will order the pairs in the order of their event information sum. Update the pair list to keep the order whenever a side event merged into a topic event. Obviously, this is again following the IFC idea. Notice that we may use a higher base threshold to decide if coreferent or not, because this step has less reliability compared to step 1.

**Algorithm 3** Topic-Side Event Clustering (TSEC) - Step 2: Topic & Side

```
Input: $\mathbf{\Gamma}_1$ (clusterings for merged documents from Step 1)
Output: $\mathbf{\Gamma}_2$ (clusterings after side events merged into topic events)
$\mathbf{\Gamma}_2 = \mathbf{\Gamma}_1$
$\mathbf{\Gamma}_2.sortTopicEvents()$
$\mathbf{\Gamma}_2.sortSideEvents()$
while $!\mathbf{\Gamma}_2.finalized()$ do
    $\mathbf{C}_{topic}, \mathbf{C}_{side} = \mathbf{\Gamma}_2.getNextUntouchedMaxInfoTopicSidePair()$
    if $calCoreSimilarity(\mathbf{C}_{topic}, \mathbf{C}_{side}) \geq \theta(\mathbf{C}_{topic}, \mathbf{C}_{side})$ then
        if $\mathbf{C}_{topic}\mathbf{C}_{side}$ from different documents then
            $\mathbf{C}_{topic}.merge(\mathbf{C}_{side})$
        end if
    end if
    $\mathbf{C}_{topic}.info = calInfoScore(\mathbf{C}_{topic})$
    $\mathbf{\Gamma}_2.reOrder(\mathbf{C}_{topic})$
    $\mathbf{\Gamma}_2.remove(\mathbf{C}_{side})$
end while
```

The algorithm is described in Algorithm 3. The key is to, again, use X+Y sorting to decide the order, where X is set to the list of topic events, and Y is set to the list of side events. A slight change is to avoid topic and side events that are from the same document. After each merge, the lists will be updated and X+Y sorting will be reset. This process is again similar to IFC.

### 7.2.3 Side Event and Side Event

The third step in TSEC attempts to make links among side events. This is the last step of TSEC algorithm. Hopefully, most of important links have already been caught by previous two steps. We can choose to use highest base threshold among the three steps since this step is the least reliable one. Most of clusters compared in this step have lower event information.

The algorithm is described in Algorithm 4. You may notice that it looks a lot like IFC, with the addition constraint that both clusters should not contain event mentions from the same document.

**Algorithm 4** Topic-Side Event Clustering (TSEC) - Step 3: Side & Side

```
**Input:** $\mathbf{\Gamma}_2$ (clustering generated from step 2)
**Output:** $\mathbf{\Gamma}_3$ (final clustering)
$\mathbf{\Gamma}_3 = \mathbf{\Gamma}_2$
$\mathbf{\Gamma}_3.sideEvents.sortByInfo()$
while $!\mathbf{\Gamma}_3.finalized()$ do
    $\mathbf{C}_1, \mathbf{C}_2 = \mathbf{\Gamma}_3.getNextUntouchedMaxInfoSidePair()$
    if $calCoreSimilarity(\mathbf{C}_1, \mathbf{C}_2) \geq \theta(\mathbf{C}_1, \mathbf{C}_2)$ then
        if $noSharedDocs(\mathbf{C}_1, \mathbf{C}_2)$ then
            $\mathbf{C} = merge(\mathbf{C}_1, \mathbf{C}_2)$
            $\mathbf{C}.info = calInfoScore(\mathbf{C})$
            $\mathbf{\Gamma}_3.sideEvents.insertByInfo(\mathbf{C})$
            $\mathbf{\Gamma}_3.sideEvents.remove(\mathbf{C}_1)$
            $\mathbf{\Gamma}_3.sideEvents.remove(\mathbf{C}_2)$
        end if
    end if
end while
```

# Chapter 8

# EVCO System

## 8.1 Overall Design

The EVCO (EVent COreference) system contains both the baseline system and the system implementing the main idea in this thesis. The system could be split into five core modules as shown in figure 8.1. Both event coreference systems share the same data reader, preprocessor, and evaluators.

## 8.2 Details

### 8.2.1 Information-based Event Coreference

The figure 8.2 is an overview of the informantion-based event coreference's algorithm part of the EVCO System's implementation.

This algorithm could still be divided into two crucial parts: SimilarityEVCO and ClusteringEVCO. SimilarityEVCO implements the Core Similarity idea mentioned in previous chapter. There are four static classifiers contained within this object, each of which is in charge of scoring the similarity for one or several kinds of event components. A trained SimilarityEVCO is used as distance metrics in ClusteringEVCO and the IFC and TSEC contained within it.

The ClusteringEVCO, which contains several IFC and one final TSEC, implements the main logic of the algorithm. Due to the fact that both IFC and TSEC use the X+Y Sorting as their backbones, we implemented a general purpose class called SortedMerge for the sorting, and pass in different commands for slightly different purpose required in IFC and TSEC.

The clustering algorithm operates on a list of ClusterEVCOs, which contains the representative event arguments and the trigger. It has the ability to perform cluster merge and also to calculate the event information score of the cluster. It also has a list of MentionEVCOs as its member. The MentionEVCO is in charge of storing information about an event mention, and also extracting the event arguments and assigning trustworthiness score to each argument. Thus, it also contains five classifiers as its static members to do

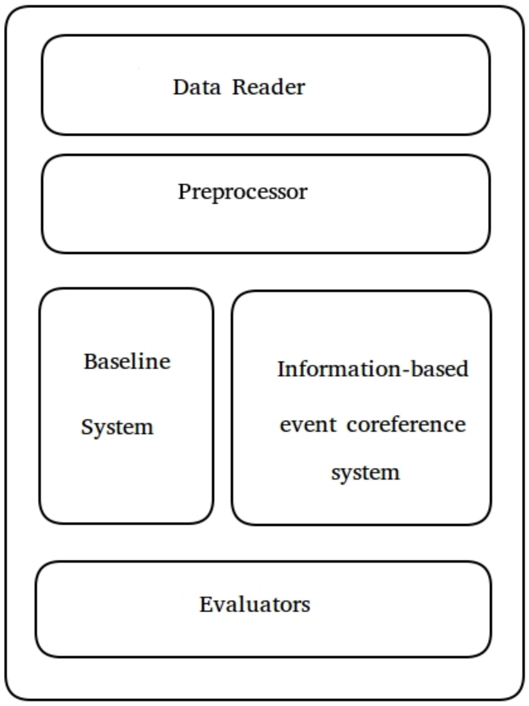

Figure 8.1: The main structure in EVCO System.

the scoring. The argument extraction is done by aligning the trigger with the illinois SRL's annotations' predicate, and then get the "AM-TMP", "AM-LOC","A0","A1" and "A2" arguments to align.

### 8.2.2 Other Core Modules

The data reader module is written to read the raw ECB data into Corpus data structure, which further stores all the data into Topics, Documents and descendent data structures until Mention level. The user is allowed to tell the program, via the configuration file, if it needs to annotate the full corpus with CachedCurator which utilizes illnois curator package to get all the annotation. The user is also allowed to specify what kind of annotations are wanted. The full corpus, along with all the annotations, could be cached for much faster data reading phase in further experiments.

The preprocessor module controls how a user uses the data. It is allowed to specify how to divide up the data to training, developing, and testing portions. The program also allows user to control how skewed the training/testing data is, by asking the user to specify the postive/negative instance ratio. This module is also in charge of transforming the data in Corpus data structure, which keeps all the information available in raw data, into instances so that classifiers defined with LBJava could directly read and use them.

The baseline system implements a simple event coreference system. The similarity metric used could be

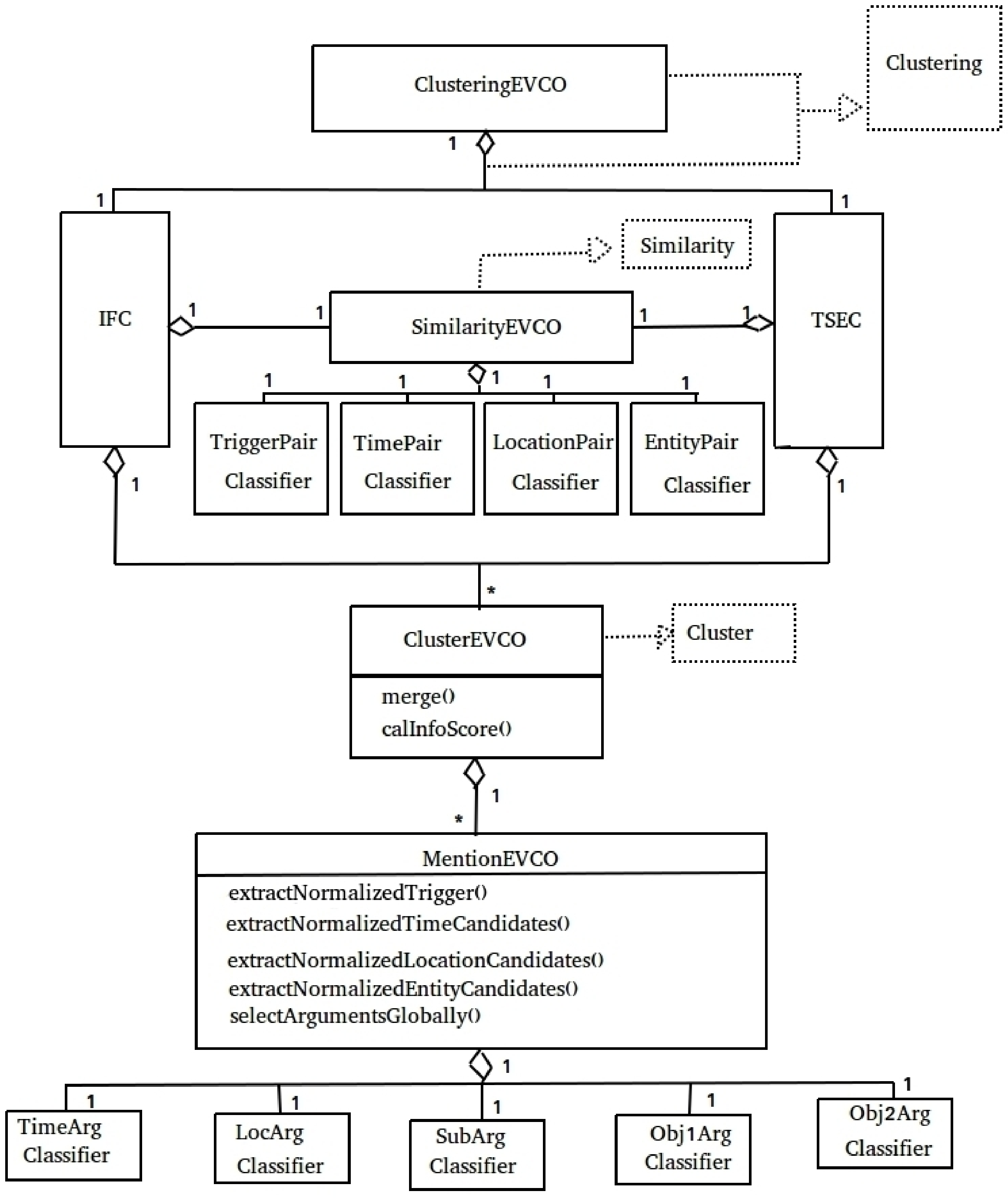

Figure 8.2: Algorithm implementation in EVCO System.

chosen from either LLM or a trained classifier. The clustering algorithm used is BestLink, which is usually in Entity Coreference tasks.

The evaluator module implements four different popular evaluation algorithms in event coreference field - MUC, B3, BLANC, and CEAF. The implementations strictly follow the original paper of these evaluation algorithms and have been carefully tested.

All of the classifiers used within this system is written and learned with the LBJava programming language, which allows for easy feature engineering and algorithm tuning.

# Chapter 9

# Experiment

## 9.1 Data

The main data used in our experiment is ECB, which is popular for both within-document event coreference and cross-document event coreference. For this data set, event mention triggers have been annotated to indicate existence of event mentions. Also, the data set's documents are clustered into topics. But we do not use this topic clustering information within our experiment. The details about this dataset is described in table 9.1.

Table 9.1: ECB Data Analysis

| | |
|---|---|
| Total Topic Count | 43 |
| Total Document Count | 482 |
| Total Annotated Mention Count | 1744 |
| Average Annotated Mentions Per Document | 3.62 |
| Total Within-Doc Coreference Mention Pair Count | 861 |
| Total Within-Doc Mention Pair Count | 4340 |
| Average Within-Doc Coreference Per Document | 1.79 |

Another data set that plays a role in these experiments is the ACE2005, another data set that is typically used in event coreference tasks. Unfortunately, ACE2005 does not come with cross document event coreference annotations. We use ACE2005 to train the event argument mention pair binary classifiers, which serve as similarity metrics for event arguments. The data is preprocessed into event argument mention pairs containing both coreference and non-coreference instances. We also use this data to train classifiers to estimate event arguments' trustworthiness scores.

## 9.2 Experiment

### 9.2.1 Core Similarity Event Argument Pairwise Classifier Performance

Our proposed Core Similarity metric contains four different event argument similarity comparators as its components. The trigger comparator is in charge of comparing trigger mentions, time comparator for time pairs, location comparator for location pairs, and entity comparator for subject pairs and object pairs. These comparators are all trained on ACE2005's within document event argument annotations.

The comparators' performance is shown in table 9.2.

Table 9.2: Event Argument Similarity Comparator Performance (ACE2005)

| Comparator | Coreference | | | Non-Coreference | | | Total |
|---|---|---|---|---|---|---|---|
| | Precision | Recall | F1 | Precision | Recall | F1 | Acurracy |
| Trigger | 92.732 | 46.015 | 61.508 | 64.100 | 96.393 | 76.998 | 71.204 |
| Entity | 72.954 | 38.104 | 50.061 | 58.113 | 85.874 | 69.317 | 61.989 |
| Time | 95.000 | 47.500 | 63.333 | 65.000 | 97.500 | 78.000 | 72.500 |
| Location | 83.333 | 26.316 | 40.000 | 56.250 | 94.737 | 70.588 | 60.526 |

The comparators' performance is relatively bad. We identify several main reasons after inspecting the data and predictions. One main reason is that, since we are mainly dealing with within-doc event argument annotations while training with ACE2005, there are a lot of pronouns and other kinds of references, which makes the direct comparison of event argument mentions much harder. And in fact, most of the error is triggered by this. Thus we need a better normalization of entities for both the entity comparator and the location comparator. Namely, a good entity coreference tool should be included into our system to preprocess the documents, which the current version does not have yet. Also, a better feature engineering could always help improve the performance in the future. Lastly, we need to notice the property of event related tasks itself, which is in fact very ambiguous so that even interannotator agreement could be pretty low in event data sets like ACE2005.

On the other hand, we do not expect the performance of these classifiers to be too high in the risk of overfitting the ACE2005 dataset, since what we will evaluate on in the end is the ECB data.

### 9.2.2 Similarity Metric Comparison

In this experiment, we compare the similarity metrics between a baseline similarity and the similarity we described in this thesis. The way we set it up is using the same clustering algorithm, here we use the EVCO system's information based algorithms (IFC and TSEC), and separately apply the baseline similarity metric and the proposed Core Similarity metric as the distance metric, and see the difference in performance. The data we experiment on is the ECB, which has been preprocessed to split into training data and evaluation data. The baseline similarity is trained with this ECB training data, while the Core Similarity metric is trained on both this ECB training data (for trigger pair comparator), and ACE2005 (for argument pair comparators). Also, due to the fact that we do not have high performance argument comparators yet, we choose not to give a high weight to their predictions, but use them more like a regularization.

We compare these two similarity metrics in both within-doc situation and cross-doc situation. The within-doc situation is evaluated over each document that was not used for training. Evaluations are generated for each document, and we calculate the average to be the final within-doc evaluations. For the cross-doc situation, we only generate one set of evaluations for the whole input data set. We randomly choose a subset of the evaluation data to be the input data due to efficiency issue.

The result is shown in table 9.3:

Table 9.3: Similarity Metric Comparison

| Model | MUC | | | B3 | | | CEAF | | | BLANC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | F1 | Precision | Recall | F1 | Precision | Recall | F1 | Precision | Recall | F1 |
| Baseline(within) | 0.129 | 0.106 | 0.110 | 0.994 | 0.940 | 0.958 | 0.918 | 0.961 | 0.933 | 0.888 | 0.889 | 0.883 |
| EVCO(within) | **0.154** | **0.135** | **0.139** | 0.973 | **0.948** | 0.951 | **0.920** | 0.940 | 0.923 | 0.880 | 0.876 | 0.873 |
| Baseline(cross) | 0.714 | 0.345 | 0.465 | 0.862 | 0.419 | 0.564 | 0.309 | 0.665 | 0.422 | 0.674 | 0.551 | 0.561 |
| EVCO(cross) | 0.591 | **0.448** | **0.510** | 0.678 | **0.447** | 0.539 | **0.363** | 0.558 | **0.440** | 0.553 | 0.543 | 0.547 |

The first thing we notice about the result is that the MUC scores are low. This is not so strange since the ECB data set's within-doc coreference rate is not that high, while MUC is a link based clustering evaluation method. That means, possibly in one ECB document, there are only two out of eight event mentions are coreferent to each other. Therefore, if the clustering algorithm missed this pair of event mentions unfortunately, MUC will punish this error heavily.

According to the results, The Core Similarity brings a little bit improvement over the baseline in MUC evaluation in the within-doc situation. It might indicate that more coreferent event mention pairs that were not picked up by baseline similarity metric, are chosen by our Core Similarity metric. In the cross-doc situation, we see significant improvement in the MUC evaluation. This is reasonable because, in cross-doc situation, event mention clusters tend to have complete information about event arguments, so that comparing based on event arguments is more reliable. More coreferent event mention clusters are detected, thus the MUC score goes up. We can also see that our Core Similarity metric gains some improvement in

CEAF score.

### 9.2.3 Clustering Algorithm Comparison

The next experiment we carry out is comparing the clustering algorithms. We compare our Information-Based clustering algorithms (IFC and TSEC) to Bestlink, the popularly used algorithm in coreference tasks. For both algorithms, we use the baseline similarity metric as the distance metric for simplicity. The results are shown in table 9.4.

Table 9.4: Clustering Algorithm Comparison

| Model | MUC | | | B3 | | | CEAF | | | BLANC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | F1 | Precision | Recall | F1 | Precision | Recall | F1 | Precision | Recall | F1 |
| BestLink(within) | 0.127 | 0.105 | 0.110 | 0.994 | 0.940 | 0.959 | 0.919 | 0.961 | 0.934 | 0.889 | 0.890 | 0.884 |
| EVCO(within) | 0.129 | 0.106 | 0.110 | 0.994 | 0.940 | 0.958 | 0.918 | 0.961 | 0.933 | 0.888 | 0.889 | 0.883 |
| BestLink(cross) | 0.714 | 0.345 | 0.465 | 0.862 | 0.419 | 0.564 | 0.309 | 0.665 | 0.422 | 0.674 | 0.551 | 0.561 |
| EVCO(cross) | 0.714 | 0.345 | 0.465 | 0.862 | 0.419 | 0.564 | 0.309 | 0.665 | 0.422 | 0.674 | 0.551 | 0.561 |

Apparently, the results are almost the same. Before saying that this result means that the comparing order does not matter, we would like to point out several other possible causes. The most possible cause could be the fact that our baseline similarity metric does not heavily base on the event arguments, but focuses more on the event trigger. The other possibility is due to the data set we are using. Our algorithm will gain more when there are more missing argument situations. Namely, within-doc event coreference would be benefit more with our algorithm presumably. But the ECB data set, as has been mentioned before, is "sparse" in the sense of within-doc event coreference. We need to mention that, our algortihm is much slower than the BestLink algorithm. This could be the tradeoff for the possible gain in within-doc event coreference.

### 9.2.4 Overall System Comparison

Lastly, we compare the overall systems' performance. The baseline system is the BestLink algorithm using the baseline similarity metric as distance, while our EVCO system is using IFC and TSEC clustering algorithms separately for within-doc and cross-doc situations, and the Core Similarity metric as the distance. The data we use is still the evaluation data from ECB. The baseline system will simply run the algorithm over the full event mention space, while our EVCO system goes in a two-step style - doing within-doc event coreference first with IFC on each document, and use TSEC for cross-doc situation over the IFC outputs.

The results are shown in table 9.5.

Table 9.5: System Comparison

| Model | MUC | | | B3 | | | CEAF | | | BLANC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | F1 | Precision | Recall | F1 | Precision | Recall | F1 | Precision | Recall | F1 |
| Baseline | 0.714 | 0.345 | 0.465 | 0.862 | 0.419 | 0.564 | 0.309 | 0.665 | 0.422 | 0.674 | 0.551 | 0.561 |
| EVCO | 0.545 | **0.414** | **0.471** | 0.672 | **0.433** | 0.527 | **0.365** | 0.561 | **0.442** | 0.545 | 0.533 | 0.536 |

As we may expect after seeing previous experiments, the EVCO system gains some improvement in MUC evaluation and CEAF evaluation. This might not be seen as significant performance gain though, because B3 and BLANC evaluation dropped correpondingly.

As a simple conclusion for these experiments, we may say that these experiments show that our proposed system gains some improvement in some aspects, but the overall improvement is not significant. But this may not be sufficient to reject our two assumption - comparing order matters, and taking missing argument into consideration is important. Better feature engineering and data preprocessing (normalization) on a data that suits this system, might give us different performance results.

# Chapter 10

# Future Work

According to the experiments, this work is still not mature enough. However, there are a few reasonable ways to improve it.

One of the most important things might be improving each event argument's similarity metric, since it is obvious that good similarity metrics directly affect the performance of clustering. More reasonable features should be added, and better tuning of parameters on each classfier should also be helpful. By looking at the results, it appears that normalization of event arguments is also important. For example, a normalized time argument could be easier to be compared. Pronouns also need to be resolved in order to identify the real entity they are referring to. The Illinois time normalization package should be used. However, notice that the data we are playing with, the ECB, does not have a Document Creation Time, so there is not reference date, which might give the time normalizer a difficult time to do the job.

Another possible improvement could be loosening the constraints applied on the similarity metric. Currently what we do when comparing two event mention clusters is select the event argument mentions that have the most event information score to represent a cluster, and then apply similarity metric on these representatives to get an estimate of similarity on cluster pairs. It might be a better choice to take into consideration all of the other event argument mentions that have been merged into this cluster. Therefore, instead of basing it on a pair of representatives, we can design a similarity metric that takes the pair of distributions of event argument mentions as input and generate a similarity score out of them. The similarity metric might also be able to update the candidate distributions. This also looks like a way to feed the event coreference predictions back to the event argument extractors to help inference the extractors' prediction event arguments. This possibly provides a way to mutually improve both event coreference and event argument extraction's results.

Another contraint that could be loosened is, instead of strictly preventing clusters from the same document chain to merge with each other when doing cross-document event coreference, we allow them to be merged, but with a much higher similarity threshold. Hopefully, this could improve the recall of within-document event coreference without losing precision so that the final accuracy goes up. By allowing for this, idealy the cross-document event coreference could also be mutually improved. If this idea is applied, then within-dcoument coreference should be tuned so that it has high precision (using a higher similarity threshold), since the recall would be caught up during the cross-document event coreference step.

There are also some other small ideas to improve the model. For example, aside from the Document Creation Date we mentioned above, other meta data of the document like the document headline could also help. Especially when using TSEC, the document headline could serve well as a seed to collect all of the event mentions belonging to the topic event, since it typically contains most of the event arguments, especially in political news articles. Unfortunately we also do not have document title in ECB.

Also, different event types might not have the same structure. This mainly differs in the number of event related objects. When adapting into specific fields, we might have the information of what types of events we want to extract. In that case, we can modify the event representation so that it adapts to different event types.

We are currently sorting based on the sum of event information of both clusters before merge. But it might make more sense if we sort according to the possible event information after merge, or the "event information gain" after a merge.

Besides this, notice that the current event information updating strategy is purely based on arguments of each event cluster, but does not take the similarities between clusters into account, which should also be considered as an important part of the event information definition. However, we might need to be clever when defining this. Because it is easy that a design considering the similarity issues will make the event information calculation operation not transitive.

Last but not least, topic modeling techniques can certainly help, especially in the case of cross-document event coreference.

# Chapter 11

# Conclusion

In conclusion, we propose a clustering algorithm that takes the order of merging event mention pairs into account, along with a similarity metric that takes care of the situation when event mentions are missing some event arguments. An event coreference system is built with these algorithms implemented. There are many possible improvements that could be done based on this preliminary work.

# References

[1] J. Araki, Z. Liu, E. Hovy, and T. Mitamura. Detecting subevent structure for event coreference resolution. *LREC 2014*, 2014.

[2] F. T. Asr, J. Sonntag, Y. Grishina, and M. Stede. Conceptual and practical steps in event coreference analysis of large-scale data. *ACL 2014*, page 35, 2014.

[3] A. Bagga and B. Baldwin. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–6. Citeseer, 1998.

[4] A. Bagga and B. Baldwin. Cross-document event coreference: Annotations, experiments, and observations. In *Proceedings of the Workshop on Coreference and its Applications*, pages 1–8. Association for Computational Linguistics, 1999.

[5] C. Bejan, M. Titsworth, A. Hickl, and S. Harabagiu. Nonparametric bayesian models for unsupervised event coreference resolution. In *Advances in Neural Information Processing Systems*, pages 73–81, 2009.

[6] C. A. Bejan. Deriving chronological information from texts through a graph-based algorithm. In *FLAIRS Conference*, pages 259–260, 2007.

[7] C. A. Bejan and S. Harabagiu. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1412–1422. Association for Computational Linguistics, 2010.

[8] C. A. Bejan and S. Harabagiu. Unsupervised event coreference resolution. 2013.

[9] C. A. Bejan and S. M. Harabagiu. A linguistic resource for discovering event structures and resolving event coreference. In *LREC*, 2008.

[10] C. Bin, S. Jian, and T. C. Lim. Resolving event noun phrases to their verbal mentions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 872–881. Association for Computational Linguistics, 2010.

[11] C. Bin, S. Jian, and T. C. Lim. A twin-candidate based approach for event pronoun resolution using composite kernel. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 188–196. Association for Computational Linguistics, 2010.

[12] K.-W. Chang, R. Samdani, and D. Roth. A constrained latent variable model for coreference resolution. In *EMNLP*, pages 601–612, 2013.

[13] B. Chen, J. Su, S. J. Pan, and C. L. Tan. A unified event coreference resolution by integrating multiple resolvers. In *IJCNLP*, pages 102–110, 2011.

[14] B. Chen, J. Su, and C. L. Tan. Random walks down the mention graphs for event coreference resolution. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(4):74, 2013.

[15] Z. Chen and H. Ji. Event coreference resolution: Algorithm, feature impact and evaluation. In *Proceedings of Events in Emerging Text Types (eETTs) Workshop, in conjunction with RANLP, Bulgaria*, 2009.

[16] Z. Chen and H. Ji. Graph-based event coreference resolution. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*, pages 54–57. Association for Computational Linguistics, 2009.

[17] Z. Chen, H. Ji, and R. Haralick. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *Proceedings of the Workshop on Events in Emerging Text Types*, pages 17–22. Association for Computational Linguistics, 2009.

[18] J. Clarke, V. Srikumar, M. Sammons, and D. Roth. An nlp curator (or: How i learned to stop worrying and love nlp pipelines). In *LREC*, pages 3276–3283, 2012.

[19] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[20] A. Cybulska and P. Vossen. Semantic relations between events and their time, locations and participants for event coreference resolution. In *RANLP*, pages 156–163, 2013.

[21] A. Cybulska and P. Vossen. Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution. In *Proceedings of LREC*, volume 2014, 2014.

[22] L. Danlos. Event coreference between two sentences. In *Computing Meaning*, pages 271–288. Springer, 2001.

[23] R. Delmonte. Coping with implicit arguments and events coreference. *NAACL HLT 2013*, page 1, 2013.

[24] R. Gaizauskas. Xi: A knowledge representation language based on cross-classification and inheritance. *Research Memorandum CS-95-24, Department of Computer Science, Univeristy of Sheffield*, 1995.

[25] R. Gaizauskas and K. Humphreys. Quantitative evaluation of coreference algorithms in an information extraction system. *Corpus-based and computational approaches to discourse anaphora*, pages 143–167, 1996.

[26] C. Girardi, M. Speranza, R. Sprugnoli, and S. Tonelli. Cromer: a tool for cross-document event and entity coreference.

[27] G. Glavaš and J. Šnajder. Exploring coreference uncertainty of generically extracted event mentions. In *Computational Linguistics and Intelligent Text Processing*, pages 408–422. Springer, 2013.

[28] K. Goyal, S. K. Jauhar, H. Li, M. Sachan, S. Srivastava, and E. H. Hovy. A structured distributional semantic model for event co-reference. In *ACL (2)*, pages 467–473, 2013.

[29] L. Hasler and C. Orasan. Do coreferential arguments make event mentions coreferential. In *Proc. the 7th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2009)*, 2009.

[30] E. Hovy, T. Mitamura, F. Verdejo, J. Araki, and A. Philpot. Events are not simple: Identity, non-identity, and quasi-identity. In *NAACL HLT*, volume 2013, page 21, 2013.

[31] J. A. E. Hovy and T. Mitamura. Evaluation for partial event coreference. *ACL 2014*, page 68, 2014.

[32] K. Humphreys, R. Gaizauskas, and S. Azzam. Event coreference for information extraction. In *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 75–81. Association for Computational Linguistics, 1997.

[33] H. Lee, M. Recasens, A. Chang, M. Surdeanu, and D. Jurafsky. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500. Association for Computational Linguistics, 2012.

[34] P. Li, Q. Zhu, and X. Zhu. A clustering and ranking based approach for multi-document event fusion. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2011 12th ACIS International Conference on*, pages 159–165. IEEE, 2011.

[35] Z. Liu, J. Araki, E. Hovy, and T. Mitamura. Supervised within-document event coreference using information propagation.

[36] X. Luo. On coreference resolution performance metrics. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 25–32. Association for Computational Linguistics, 2005.

[37] K. McConky, R. Nagi, M. Sudit, and W. Hughes. Improving event co-reference by context extraction and dynamic feature weighting. In *Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2012 IEEE International Multi-Disciplinary Conference on*, pages 38–43. IEEE, 2012.

[38] H. Peng, D. Khashabi, and D. Roth. Solving hard coreference problems. *Urbana*, 51:61801, 2015.

[39] S. Pradhan, L. Ramshaw, M. Marcus, M. Palmer, R. Weischedel, and N. Xue. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27. Association for Computational Linguistics, 2011.

[40] V. Punyakanok, D. Roth, and W.-t. Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287, 2008.

[41] M. Recasens and E. Hovy. Blanc: Implementing the rand index for coreference evaluation. *Natural Language Engineering*, 17(04):485–510, 2011.

[42] N. Rizzolo and D. Roth. Learning based java for rapid development of nlp systems. In *LREC*, 2010.

[43] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[44] E. Tomadaki and A. Salway. Matching verb attributes for cross-document event co-reference. In *Procs. Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes*, pages 127–132, 2005.

[45] M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*, pages 45–52. Association for Computational Linguistics, 1995.

[46] R. Zhao, Q. X. Do, and D. Roth. A robust shallow temporal reasoning system. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstration Session*, pages 29–32. Association for Computational Linguistics, 2012.