Computing Cooperation in Annotated Dialogues

Brian Plüss

This document describes the algorithms for automatically computing conversational cooperation of a dialogue, given an annotated transcript and a dialogue game.

Linguistic cooperation of a dialogue participant with respect to a conversational setting equates to the participant following the rules of the dialogue game for that conversational setting. From this perspective, each turn in a dialogue is associated with an amount of cooperation and an amount of non-cooperation, given by the number of dialogue rules, respectively, conformed with and violated in the turn. The instances in which rules are conformed with are called **cooperative features** and those in which rules are broken are called **non-cooperative features**:

> Participants can break the rules of the game in two ways: (a) by performing a conversational action that is not allowed for their role and (b) by failing to perform an action they were obliged to perform. Instances of (a) are violations of static obligations, which we call **static non-cooperative features**. Instances of (b) are violations of dynamic obligations, which we call **dynamic non-cooperative features**. An analogous distinction is made for cooperative features, called, respectively, **static cooperative features** and **dynamic cooperative features**. The **degree of cooperation** of each dialogue participant is thus the ratio between the number of cooperative features – static and dynamic – and the total number of features of that participant. In general, this value can be obtained for the entire conversation and for any continuous fragments.

In the rest of the section, we revisit the formalisation of these concepts and describe a method to compute the dynamic obligations of participants in each turn of a dialogue. We also explain how to compute static and dynamic cooperative and non-cooperative features and the degree of non-cooperation.

Computing Dynamic Obligations

Formally, for a dialogue $D = \langle t_1; \ldots; t_n \rangle$, where each t_i is a turn, we represent dynamic obligations as two sequences $PO_D = \langle PO_{D,0}; PO_{D,1}; \ldots; PO_{D,n} \rangle$ and $DO_D = \langle DO_{D,1}; \ldots; DO_{D,n} \rangle$, where each element is a list $\langle o_1, \ldots, o_k \rangle$

of the **obligations pending** after and **discharged** in turn t_i, respectively. $PO_{D,0}$ is also a list with the obligations pending before the dialogue starts. As in Chapter **??**, each obligation is a pair $o_i = (s_i, l_i)$, where s_i is a speaker and l_i is an action label.

Obligations are **updated** in each turn of the dialogue. This means that, for each turn t_i, $PO_{D,i}$ and $DO_{D,i}$ are computed based on $PO_{D,(i-1)}$ by **discharging** existing obligations and **introducing** new pending ones[1]:

- A pending obligation $o = (s, l) \in PO_{D,i-1}$ is **discharged** in turn $t_i = (s_i, L_i)$ if $s = s_i$ and $l_j \succ l$ for some label $l_j \in L_i$. The resulting list of pending obligations $PO_{D,i}$ is $PO_{D,i-1} - \langle o \rangle$. The obligation is added to the list of discharged obligations for the turn so $DO_{D,i}$ becomes $DO_{D,i} \circ \langle o \rangle$. To discharge several obligations in the same turn, we generalise this definition in the obvious way.

- An obligation o is **introduced** in turn $t_i = (s_i, L_i)$ if there is a rule $[(s, l) \rightsquigarrow o]$ in the game such that $s = s_i$ and $l_j \succ l$, for some label $l_j \in L_i$ (meaning that obligations are introduced by implicitly performed actions). After the update, the list of pending obligations is $\langle o \rangle \circ PO_{D,i}$. The list of discharged obligations remains unchanged. To introduce several obligations in the same turn, we generalise this definition in the obvious way.

Those obligations in $PO_{D,(i-1)}$ that are not discharged in turn t_i are carried over to $PO_{D,(i)}$.

Algorithm 1 describes the procedure for automatically computing the sequences $PO_D = \langle PO_{D,1}; \ldots; PO_{D,n} \rangle$ and $DO_D = \langle DO_{D,1}; \ldots; DO_{D,n} \rangle$ of pending and discharged dynamic obligations for a dialogue D and starting obligations $PO_{D,0}$, given dialogue game G. The implementation of this algorithm is actually more complicated as it requires that responsive implicitly performed actions, such as acceptances and rejections, be matched to the actions they refer to. The criterion we followed considers any actions that need be accepted or rejected by the current speaker and for which the corresponding obligations were introduced in the turn immediately preceding the current one. The implicit action performed with respect to such actions is then determined by the rules in the set *Discharge* of the dialogue game[2].

[1] In the definition below, recall that the implicit performance relation $\succ$ is reflexive, so for any label l it is $l \succ l$. Also, in the rest of the presentation, we assume that list structures support operations for indexing (denoted as $l[i]$), concatenation ($l_1 \circ l_2$), subtraction ($l_1 - l_2$) and those inherited from sets, such as element membership ($x \in l$) and cardinality ($|l|$).

[2] Explicitly discharged obligations are straightforward to deal with by following the unification-like binding of variables and constants discussed earlier and bearing in mind that the performance relation, $\succ$, is reflexive.

Algorithm 1 Computing dynamic obligations for dialogue D, initial pending obligations $PO_D[0]$ and dialogue game G.

for i **in** $\{1 \ldots \mathsf{length}(D)\}$ **do** `[for each turn in the dialogue...]`

  $(speaker, labels) \leftarrow D[i]$ `[take actions in current turn]`
  $pending\text{-}obligations \leftarrow PO_D[i-1]$ `[take previous pending obligations]`

`[introduce new obligations]`

  for $label$ **in** $labels$ **do** `[for each action in the turn...]`
    for $rule$ **in** $G(2)$ **do** `[for each rule in the set Introduce...]`
      $[(rule\text{-}speaker, rule\text{-}label) \rightsquigarrow obligation] \leftarrow rule$
      if $(speaker = rule\text{-}speaker) \wedge (label \succ rule\text{-}label)$ **then**
        $pending\text{-}obligations \leftarrow \langle obligation \rangle \circ pending\text{-}obligations$
      end if
    end for
  end for

`[discharge obligations met in turn]`

  $discharged\text{-}obligations \leftarrow \langle\rangle$
  for $label$ **in** $labels$ **do** `[for each action in the turn...]`
    for $obligation$ **in** $pending\text{-}obligations$ **do**
      $(obligation\text{-}speaker, obligation\text{-}label) \leftarrow obligation$
      if $(speaker = obligation\text{-}speaker) \wedge (label \succ obligation\text{-}label)$ **then**
        $pending\text{-}obligations \leftarrow pending\text{-}obligations - \langle obligation \rangle$
        $discharged\text{-}obligations \leftarrow discharged\text{-}obligations \circ \langle obligation \rangle$
      end if
    end for
  end for

  $PO_D[i] \leftarrow pending\text{-}obligations$ `[set obligations pending after turn]`
  $DO_D[i] \leftarrow discharged\text{-}obligations$ `[set obligations discharged in turn]`
end for

Computing Cooperative and Non-Cooperative Features

As introduced in the previous chapter, cooperative and non-cooperative features are instances of, respectively, observed and neglected static and dynamic obligations:

Static cooperative features: actions performed by the speaker that are allowed for by his or her role in the dialogue.

Static non-cooperative features: actions performed by the speaker that are disallowed for by his or her role in the dialogue.

Dynamic cooperative features: obligations on the speaker that were discharged in the current turn.

Dynamic non-cooperative features: obligations on the speaker that were not discharged in the current turn.

Formally, for a dialogue $D = \langle t_1; \ldots; t_n \rangle$ features will be grouped in two sequences, $SF_D = \langle sf_1; \ldots; sf_n \rangle$ and $DF_D = \langle df_1; \ldots; df_n \rangle$, of static and dynamic features, respectively. The elements in both sequences are triples (s_i, C_i, NC_i) where s_i is the speaker in turn t_i, C_i is the list of cooperative features and NC_i is the list of non-cooperative features (static for sf_i and dynamic for df_i). In the rest of the section we show how compute these features.

Computing Static Features. In each turn, we check whether the actions performed by the speaker are allowed for his or her role as specified in the dialogue game. If an action is in the the speaker's set of allowed actions, then it constitutes a static cooperative feature, otherwise it becomes a static non-cooperative feature. Formally, this means that in turn $t_i = (s_i, L_i)$, for each $l \in L_i$ we check whether $l \in L$, with $[s_i : L]$ a dialogue game rule in $G(1) = \mathit{Allow}$. If this is the case, then l is added to C_i in $sf_i = (s_i, C_i, NC_i)$. Otherwise, l is added to NC_i. Algorithm 2 shows the procedure for computing static features in a dialogue D given a dialogue game G.

Computing Dynamic Features. In each turn, we look at the speaker's obligations pending after and discharged in that turn. If an obligation on the speaker has been discharged within the turn, then it constitutes a dynamic cooperative feature, otherwise it becomes a dynamic non-cooperative feature. Formally, this means that for turn $t_i = (s_i, L_i)$, an obligation $o = (s_o, l_o) \in DO_{D,i}$ discharged in the current turn for which $s_o = s_i$ is the current speaker is a dynamic cooperative feature. The action label l_o is thus added to the list C_i of dynamic cooperative features in $df_i = (s_i, C_i, NC_i)$. On the other hand, a pending obligation $o = (s_o, l_o) \in PO_{D,i}$, not discharged in the current turn for which $s_o = s_i$ is the turn speaker is a dynamic non-cooperative feature. The action label l_o is thus added to NC_i of dynamic non-cooperative features in $df_i = (s_i, C_i, NC_i)$. Algorithm 3 shows a procedure for computing dynamic features in dialogue D, given pending dynamic obligations PO_D and discharged dynamic obligations DO_D.

Computing the Degree of Non-Cooperation

Once we have computed the static and dynamic features for each turn, we can regard the proportion of these that are cooperative as an indicator of the extent to which each participant acted within the rules of the game. This is the **degree of cooperation** of a dialogue participant with respect to a dialogue game. Formally, for speaker s and dialogue $D = \langle t_1; \ldots; t_n \rangle$

Algorithm 2 Computing static features for dialogue D and game G.

for i **in** $\{1 \ldots \mathsf{length}(D)\}$ **do** `[for each turn in the dialogue...]`

$(speaker, labels) \leftarrow D[i]$ `[take current turn]`

`[collect actions allowed for the speaker]`

$allowed\text{-}actions \leftarrow \{\}$
for $rule$ **in** $G(1)$ **do** `[for each rule in the set Allow...]`
$[rule\text{-}speaker : rule\text{-}labels] \leftarrow rule$
if $speaker = rule\text{-}speaker$ **then**
$allowed\text{-}actions \leftarrow allowed\text{-}actions \cup rule\text{-}labels$
end if
end for

`[compute static features for current turn]`

$cooperative \leftarrow \langle\rangle$
$non\text{-}cooperative \leftarrow \langle\rangle$
for $label$ **in** $labels$ **do** `[for each action in the turn...]`
if $label.name \in allowed\text{-}actions$ **then**
$cooperative \leftarrow cooperative \circ \langle label \rangle$
else
$non\text{-}cooperative \leftarrow non\text{-}cooperative \circ \langle label \rangle$
end if
end for

`[set static features for current turn]`

$SF_D[i] \leftarrow (speaker, cooperative, non\text{-}cooperative)$
end for

this is:

$$dc_{D,s} = \frac{cf_{D,s}}{cf_{D,s} + ncf_{D,s}}$$

where $cf_{D,s}$ is the number of cooperative features – both static and dynamic – of participant s and $ncf_{D,s}$ is the analogous for non-cooperative features. This is[3]:

$$cf_{D,s} = \sum_{\substack{i=1 \\ [s_i = s]}}^{n} |sf_i(2)| + |df_i(2)|$$

$$ncf_{D,s} = \sum_{\substack{i=1 \\ [s_i = s]}}^{n} |sf_i(3)| + |df_i(3)|$$

[3]Recall that the elements in the sequences of both static and dynamic features $SF_D = \langle sf_1; \ldots; sf_n \rangle$ and $DF_D = \langle df_1; \ldots; df_n \rangle$ are triples (s_i, C_i, NC_i), where s_i is the speaker in turn t_i, and C_i and NC_i are the associated sequences of, respectively, cooperative and non-cooperative features.

Note that, although these definitions are here expressed for the complete dialogue, the same applies to any contiguous subsequences of turns.

The **degree of non-cooperation** of a dialogue participant s in dialogue D is:

$$dnc_{D,s} = 1 - dc_{D,s} = \frac{ncf_{D,s}}{cf_{D,s} + ncf_{D,s}}$$

Algorithm 3 Computing dynamic features for dialogue D, given pending dynamic obligations PO_D and discharged dynamic obligations DO_D.

for i **in** $\{1 \ldots \mathsf{length}(D)\}$ **do** `[for each turn in the dialogue...]`

  $(speaker, label) \leftarrow D[i]$ `[take speaker of current turn]`

  `[compute dynamic features for current turn]`

  $cooperative \leftarrow \langle\rangle$
  $non\text{-}cooperative \leftarrow \langle\rangle$

  `[collect obligations on speaker met in current turn]`

  $met\text{-}obligations \leftarrow DO_D[i]$
  for $obligation$ **in** $met\text{-}obligations$ **do**
    $(obligation\text{-}speaker, obligation\text{-}label) \leftarrow obligation$
    $cooperative \leftarrow cooperative \circ \langle obligation\text{-}label \rangle$
  end for

  `[collect obligations on speaker pending after current turn]`

  $unmet\text{-}obligations \leftarrow PO_D[i]$
  for $obligation$ **in** $unmet\text{-}obligations$ **do**
    $(obligation\text{-}speaker, obligation\text{-}label) \leftarrow obligation$
    if $obligation\text{-}speaker = speaker$ **then**
      $non\text{-}cooperative \leftarrow non\text{-}cooperative \circ \langle obligation\text{-}label \rangle$
    end if
  end for

  `[set dynamic features for current turn]`

  $DF_D[i] \leftarrow (speaker, cooperative, non\text{-}cooperative)$
end for