Module 11: Machine Learning

Supplementary material

Relation learning in GATE

Outline

The GATE approach to learning relations

How we represent relations

How we represent machine learning instances

How we represent and create features of these instances

Configuring the Batch Learning PR

Hands-on – a working toy example, and exercises

What is a relation?

A connection between combinations of things in text

Semantic relations

- "Executive VP of Goldman Sachs, Fabrice Tourre"

Grammatical relations

- e.g. dependency relations may be learned

Negation

- "definitely no sign of any tumour"

Coreference

- coreferents can be considered as related to each other

Interactions e.g. protein-protein

Intra-sentential and inter-sentential relations

etc etc – novel uses?

GATE support for relation learning

We use the Batch Learning PR

i.e. supervised learning of relations

A similar approach to that taken with entity learning and classification

The main tasks are to

- Create learning instances from relations between annotations
- Create instance attributes from annotations and their features

These tasks are configured from an xml file

The approach

Gather examples of the relation for supervised learning

Create positive and negative instances of the relation

Define instance attributes in terms of annotations and their features

Train a model

Apply the model to unseen texts

Representing relations

Until recently, GATE had no special support for relation annotation. We now have the ontology-based RAT (Relation Annotation Tool), but we have not yet used its annotations with machine learning, so this approach may change in the future.

For machine learning in GATE, we represent relations by conventions illustrated in the following slides.

- A relation is represented by a single annotation
- Relation arguments are each represented by a single annotation
- Each argument annotation has a feature with a value giving its ID
  - this is not the same as GATE's Annotation ID
- The relation annotation has two features with values giving the IDs for its arguments
- Any name can be used for annotations and features
- The relation annotation can take any span, but for visualisation, it often spans from the start of the leftmost to the end of the rightmost

Representing relations

The $1 billion fraud case brings charges against Fabrice Tourre, senior VP of Goldman Sachs

Person

Representing relations

The $1 billion fraud case brings charges against Fabrice Tourre, senior VP of Goldman Sachs

Person

Organization

Representing relations

The $1 billion fraud case brings charges against Fabrice Tourre, senior VP of Goldman Sachs

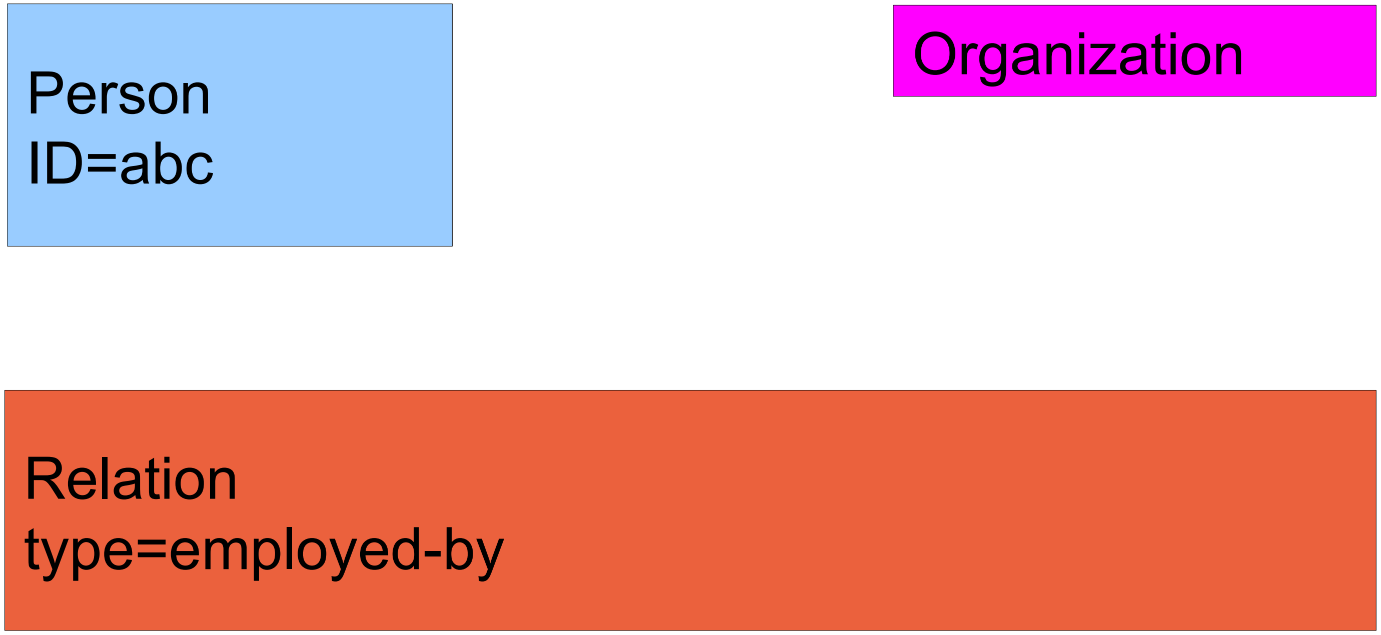
Person

Organization

Relation
type=employed-by

Representing relations

The $1 billion fraud case brings charges against Fabrice Tourre, senior VP of Goldman Sachs



Representing relations

The $1 billion fraud case brings charges against Fabrice Tourre, senior VP of Goldman Sachs

Person
ID=abc

Organization
ID=xyz

Relation
type=employed-by

Representing relations

The $1 billion fraud case brings charges against Fabrice Tourre, senior VP of Goldman Sachs

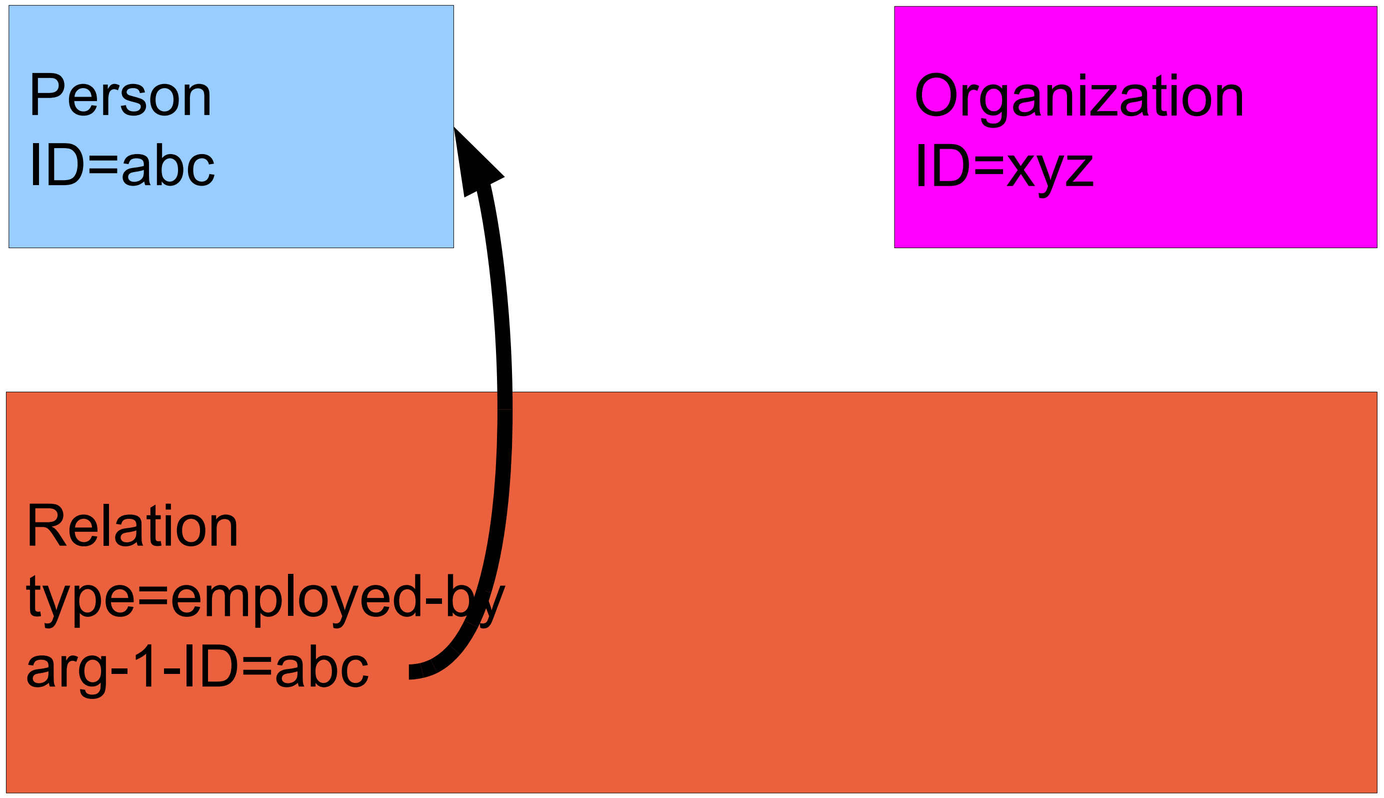
Person
ID=abc

Organization
ID=xyz

Relation
type=employed-by
arg-1-ID=abc

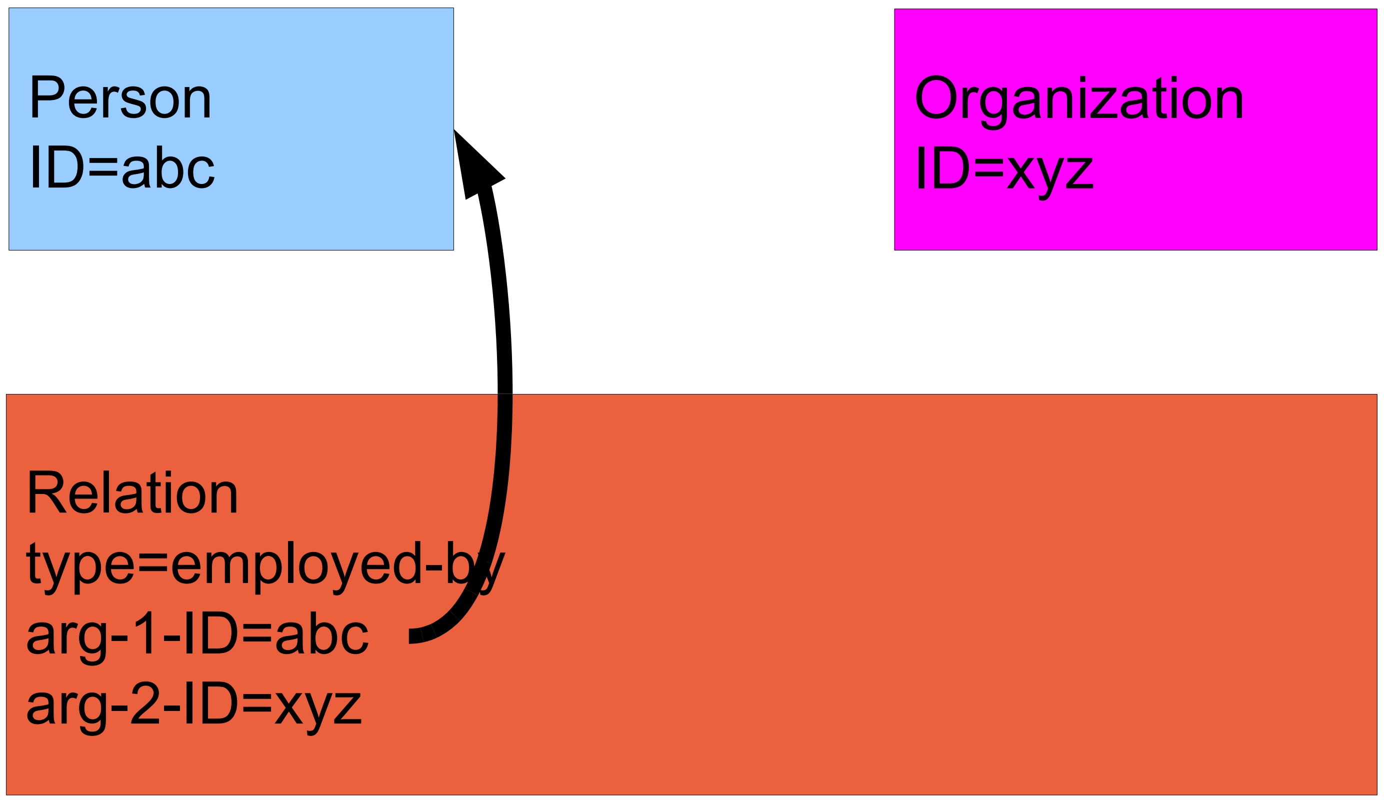
Representing relations

The $1 billion fraud case brings charges against Fabrice Tourre, senior VP of Goldman Sachs



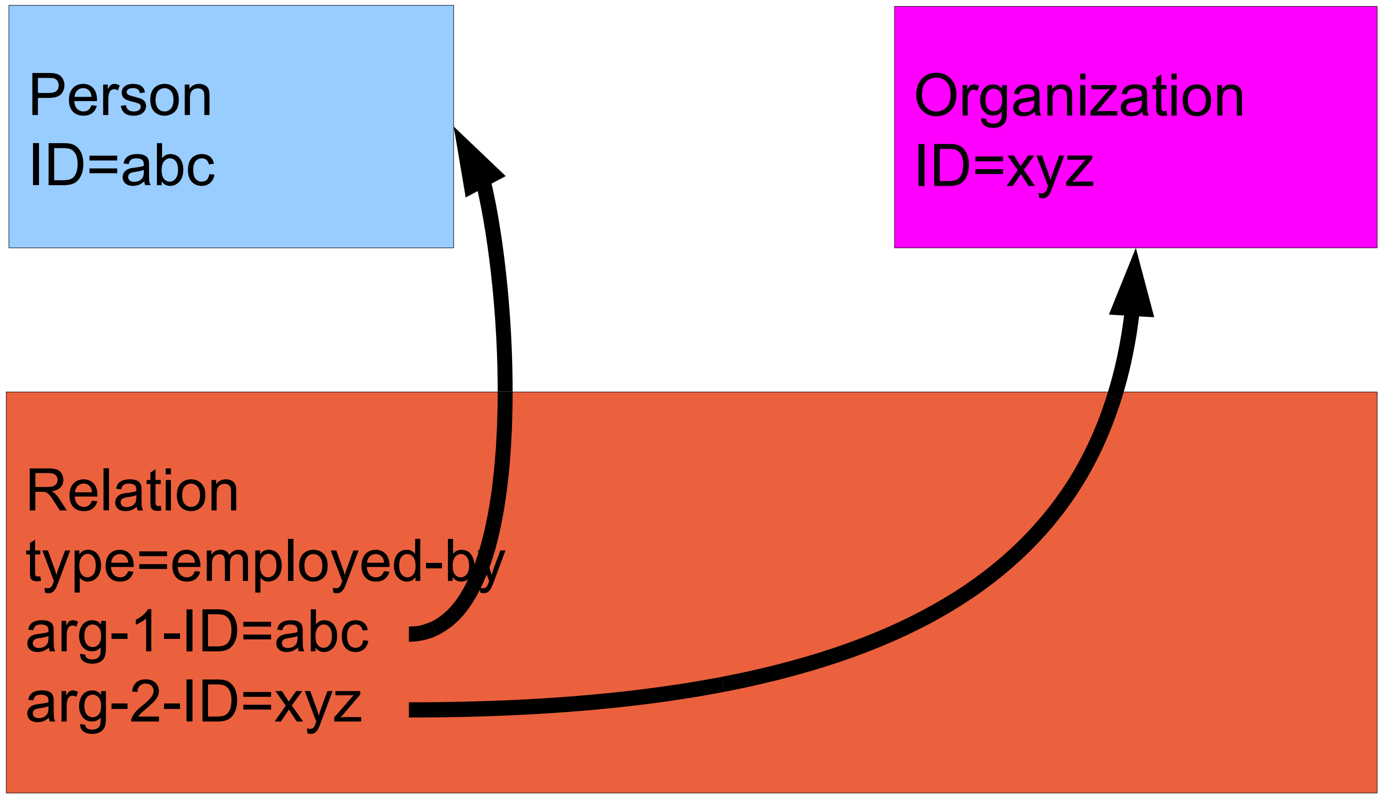
Representing relations

The $1 billion fraud case brings charges against Fabrice Tourre, senior VP of Goldman Sachs



Representing relations

The $1 billion fraud case brings charges against Fabrice Tourre, senior VP of Goldman Sachs



Relations and instances

In most cases, you will not have a straightforward classification problem with positive and negative training instances

You will have relations of one or more types marked in text, with no negative examples

We can create positive and negative training instances by considering all pairings of possible relation arguments, e.g. all entities in the same sentence

The following slides look at an example training text, and adds instances

Relations and instances

In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

Relations and instances

Person

In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

Person

Relations and instances

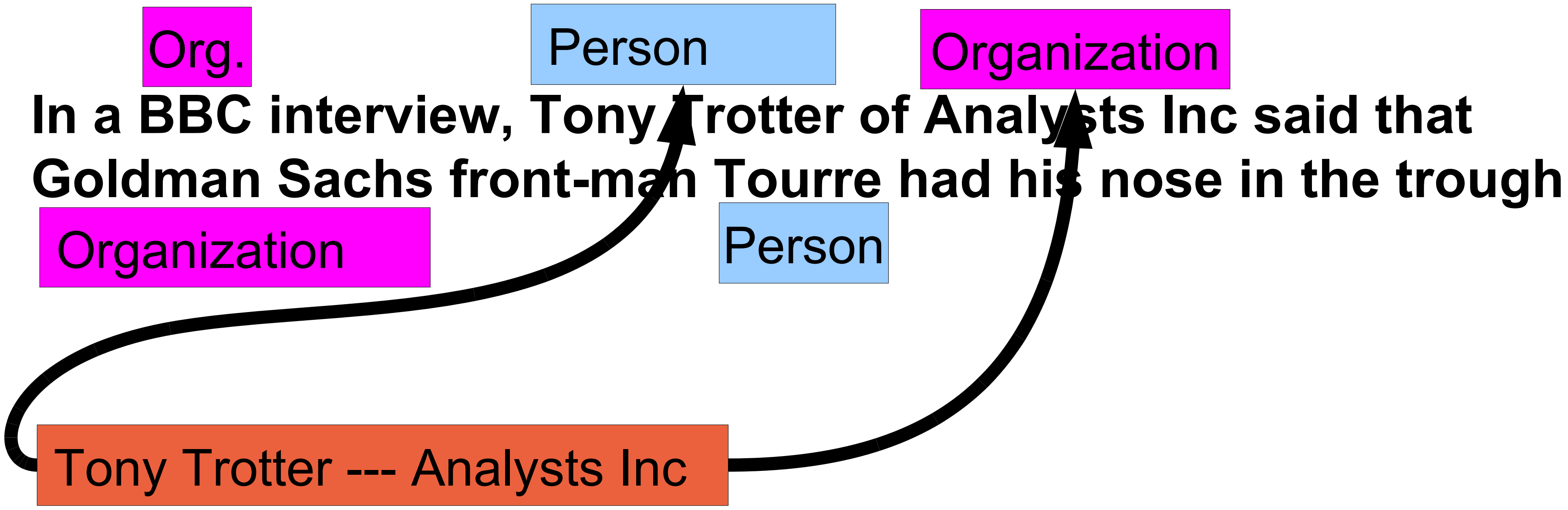
Org. Person Organization

In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

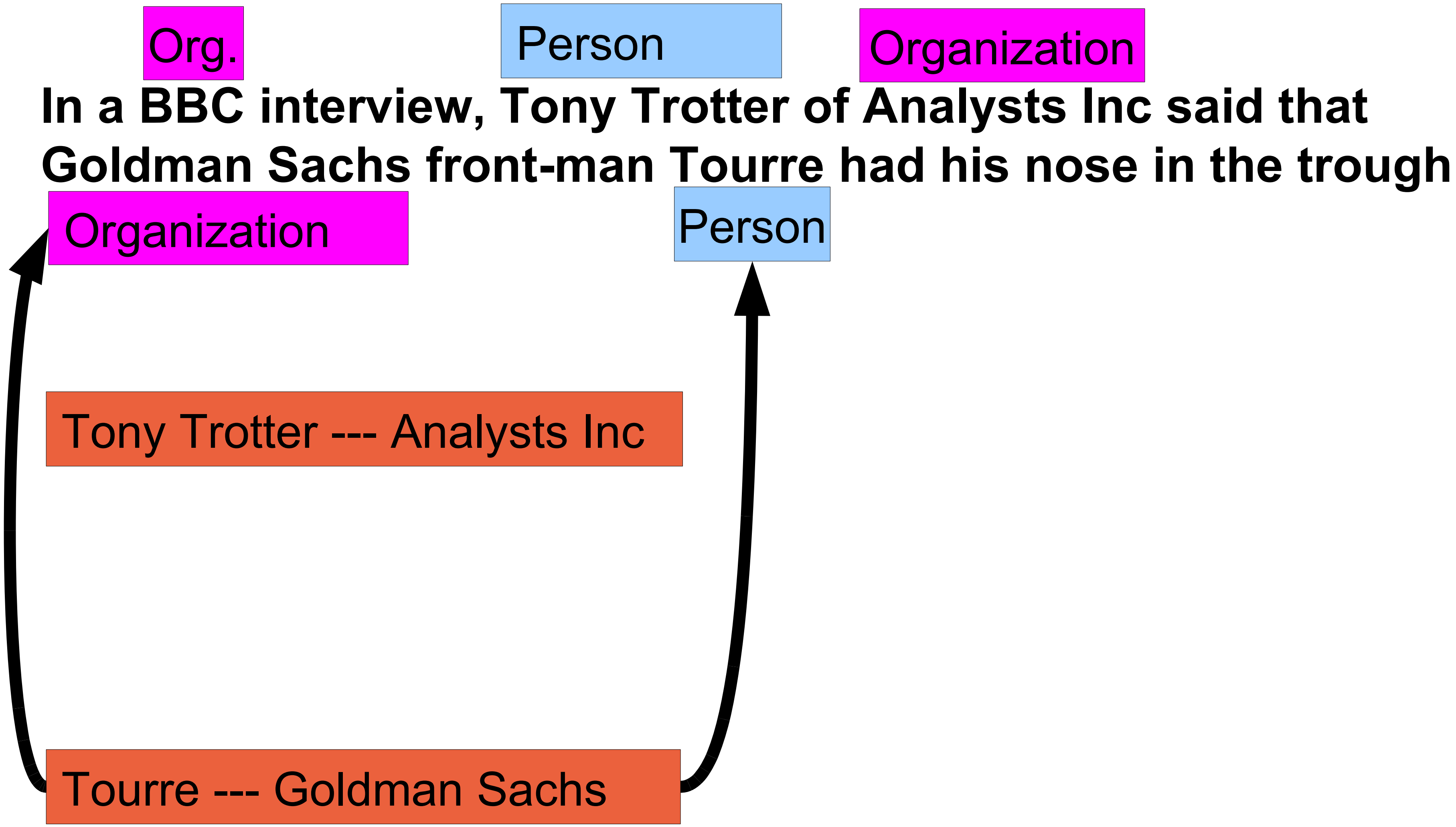
Organization Person

GATE

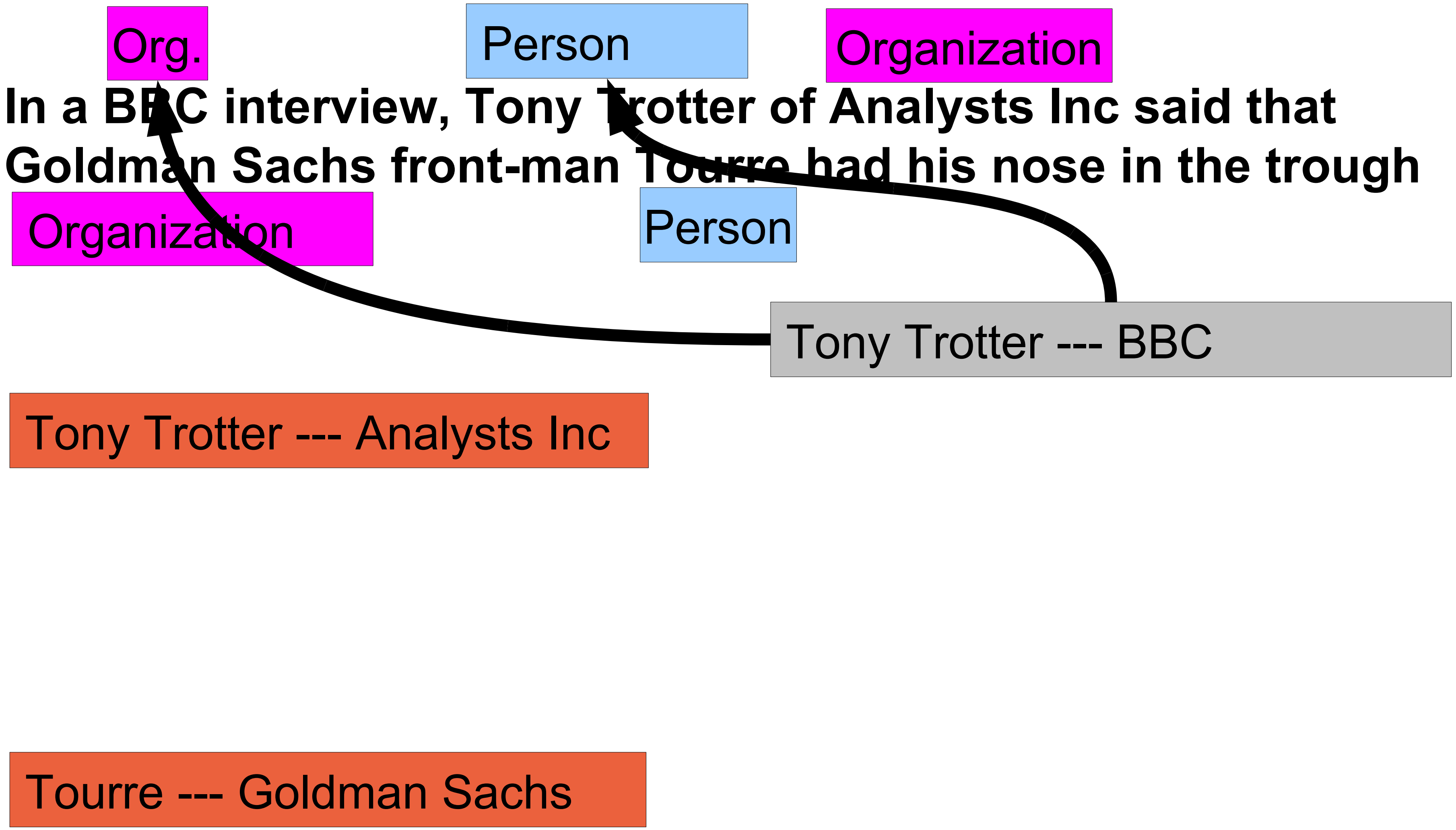
Relations and instances


Org.
Person
Organization
In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough
Organization
Person
Tony Trotter --- Analysts Inc

Relations and instances

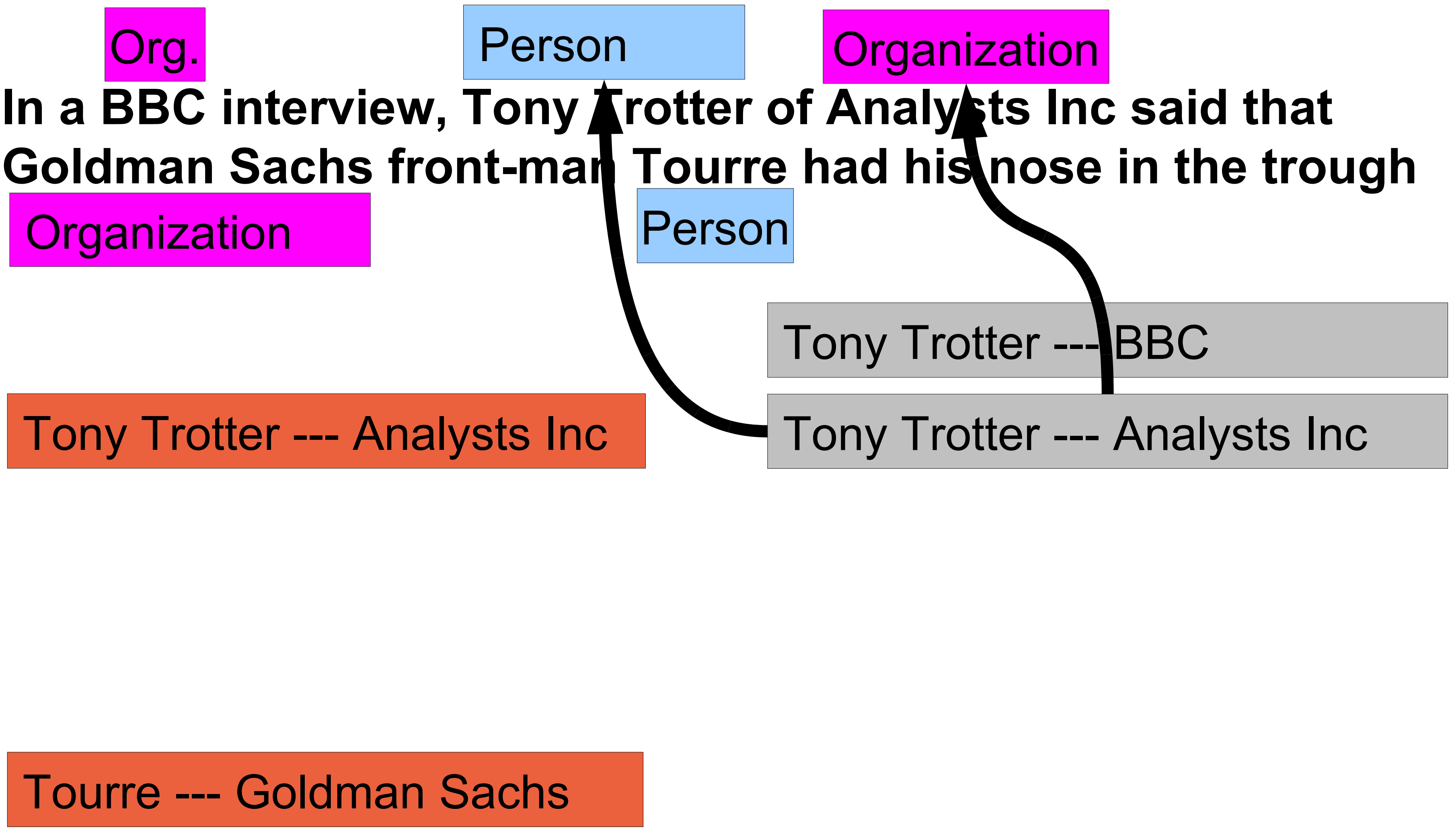

Org.
Person
Organization
In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough
Organization
Person
Tony Trotter --- Analysts Inc
Tourre --- Goldman Sachs

Relations and instances



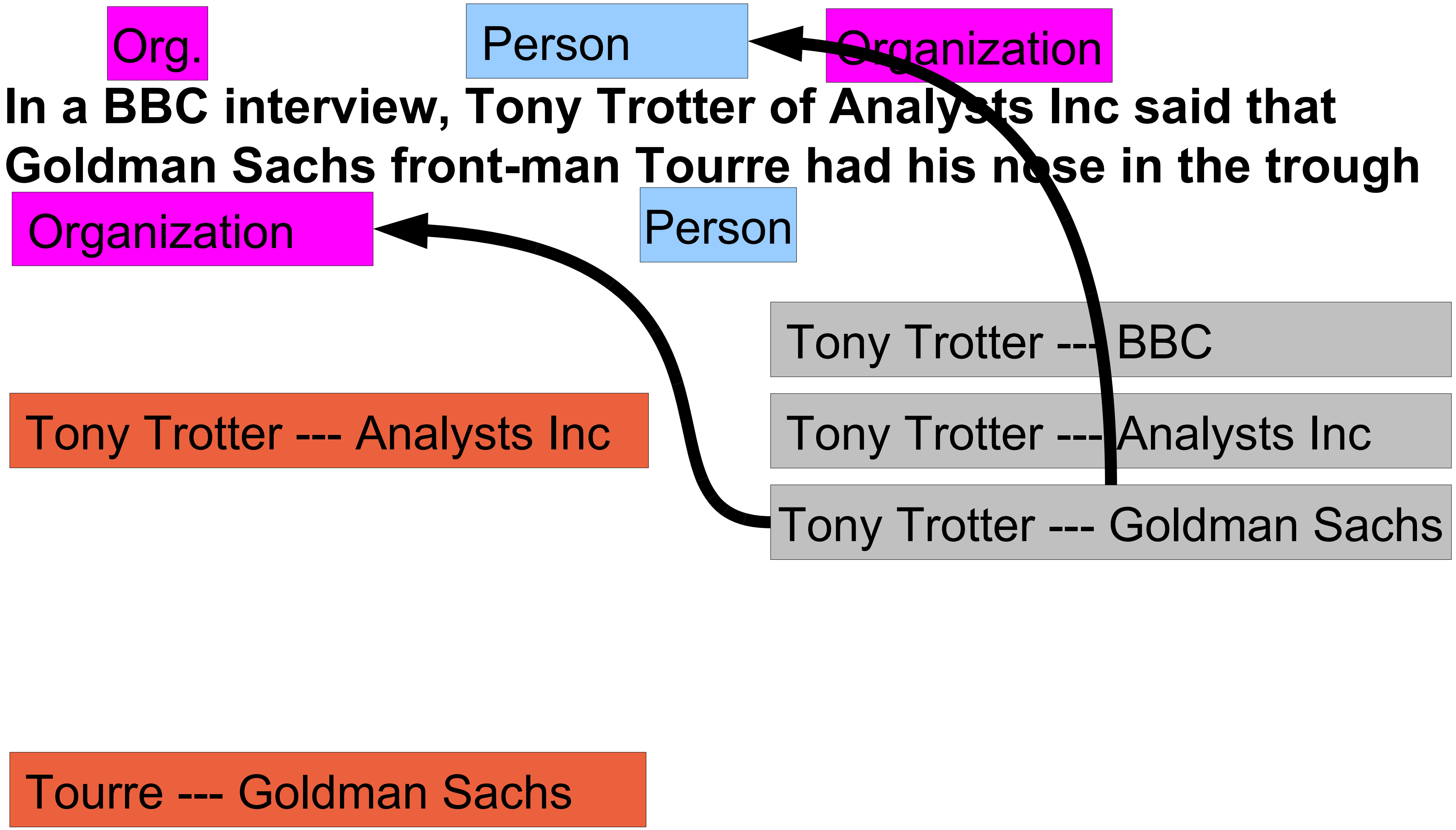
In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

Relations and instances



In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

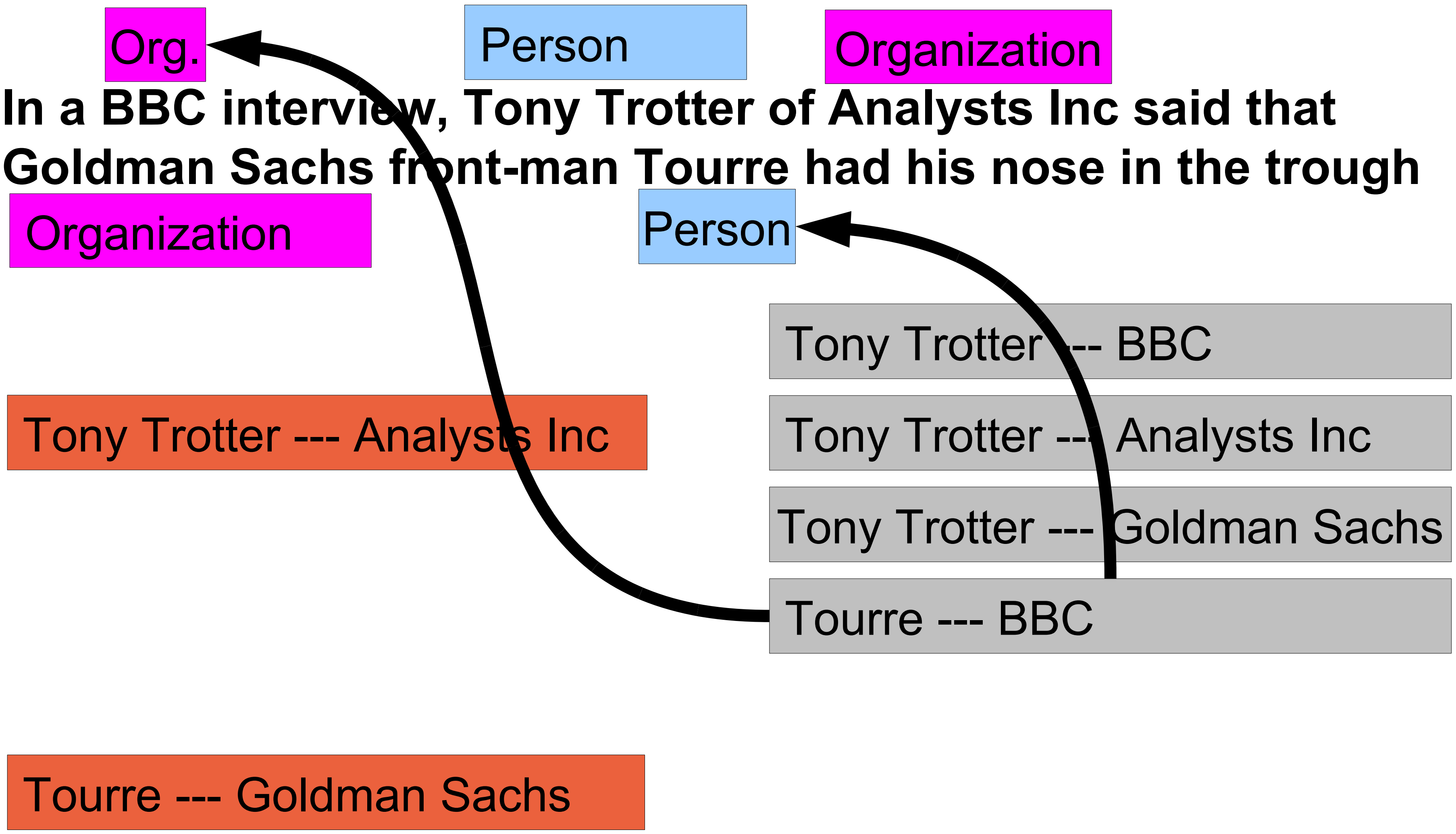
Relations and instances



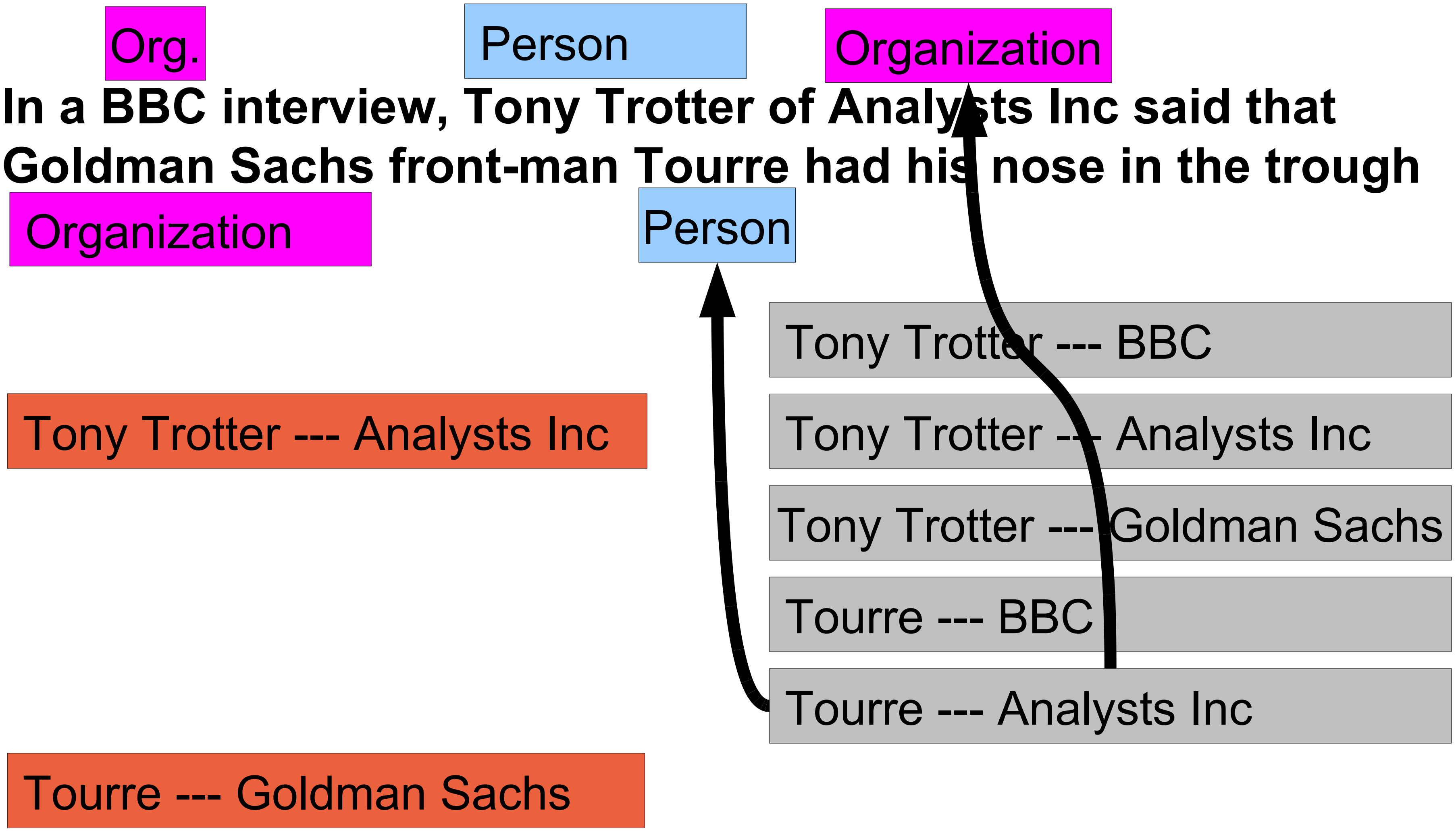
In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

Org.
Person
Organization
Organization
Person
Tony Trotter --- BBC
Tony Trotter --- Analysts Inc
Tony Trotter --- Goldman Sachs
Tony Trotter --- Analysts Inc
Tourre --- Goldman Sachs

Relations and instances

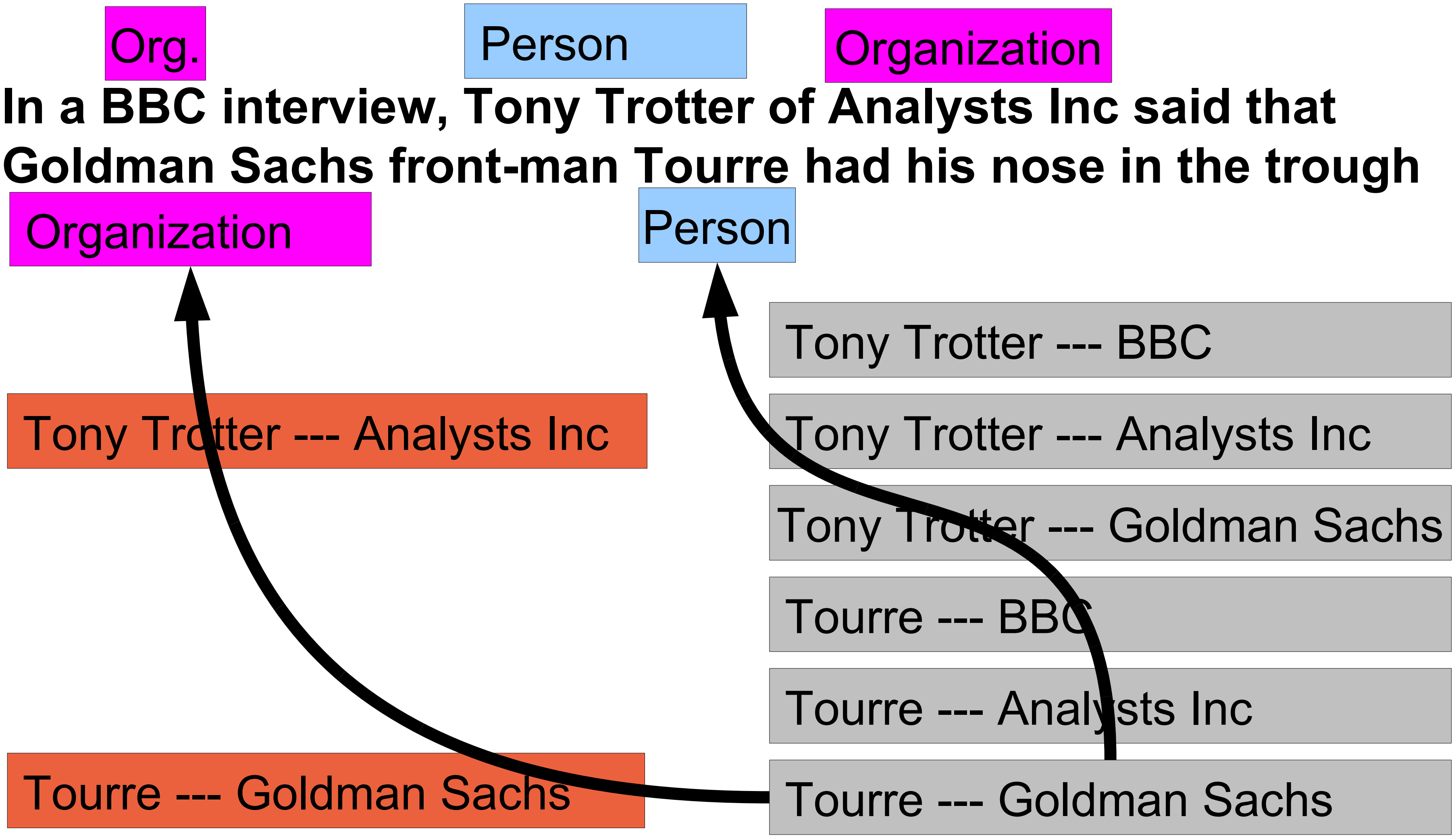

In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

Relations and instances



In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

Relations and instances


Org.
Person
Organization
In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough
Organization
Person
Tony Trotter --- BBC
Tony Trotter --- Analysts Inc
Tony Trotter --- Goldman Sachs
Tourre --- BBC
Tourre --- Analysts Inc
Tourre --- Goldman Sachs
Tony Trotter --- Analysts Inc
Tourre --- Goldman Sachs

Relations and instances

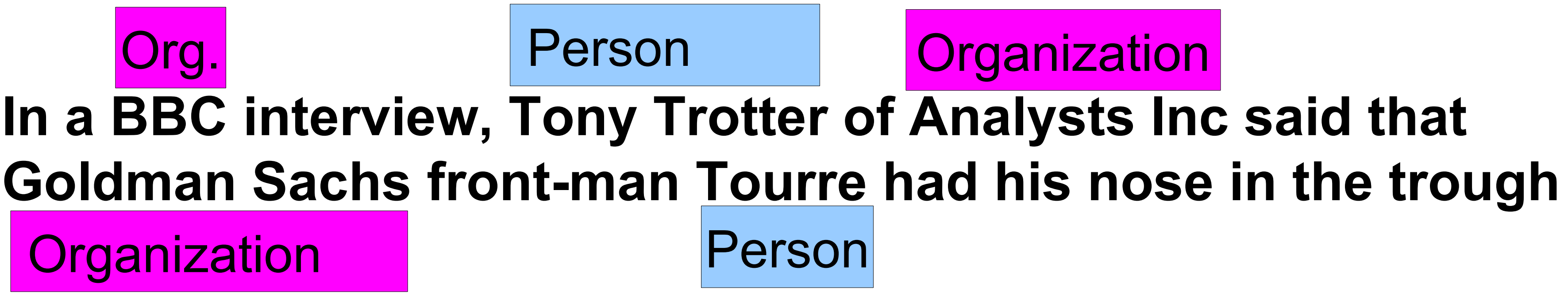
Org. Person Organization

In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

Organization Person

	Tony Trotter --- BBC
Tony Trotter --- Analysts Inc	Tony Trotter --- Analysts Inc
	Tony Trotter --- Goldman Sachs
	Tourre --- BBC
	Tourre --- Analysts Inc
Tourre --- Goldman Sachs	Tourre --- Goldman Sachs

Relations and instances



In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

	−	Tony Trotter --- BBC
Tony Trotter --- Analysts Inc		Tony Trotter --- Analysts Inc
	−	Tony Trotter --- Goldman Sachs
	−	Tourre --- BBC
	−	Tourre --- Analysts Inc
Tourre --- Goldman Sachs		Tourre --- Goldman Sachs

Relations and instances

Org. — Person — Organization

In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

Organization — Person

	−	Tony Trotter --- BBC
Tony Trotter --- Analysts Inc	+	Tony Trotter --- Analysts Inc
	−	Tony Trotter --- Goldman Sachs
	−	Tourre --- BBC
	−	Tourre --- Analysts Inc
Tourre --- Goldman Sachs	+	Tourre --- Goldman Sachs

Using instances in training and application

How do we use this in practice?

Take training examples with annotated relations

Add instances by pairing all arguments

This gives both positive and negative examples

Configure the Batch Learning PR for these annotation types

Train your model

Take application texts

Add instances by pairing all arguments

Apply the model

The Batch Learning PR will add relations where it predicts an instance is positive

Creating instances

There is no PR in the GATE distribution that will pair arguments and create instances for you

You must therefore create instances that are relevant to your problem

This will involve writing a custom PR, or some JAPE with Java RHS

Multi-class problems

The above example assumed that we had a single type of relation, ***employed-by***

When classifying instances, the Batch Learning PR chooses between the ***employed-by*** relation in the case of +ve instances, and the ***null*** relation in the case of -ve instances.

Most relation problems will be multi-class, e.g. ***employed-by***, ***fired-from***, ***hired-by***...

These relations will most likely be represented by a feature on a single annotation type (e.g. a feature ***relation-type*** on a ***Relation*** annotation)

As with entity learning, the Batch Learning PR will represent this multi-class problem as multiple binary problems, and deal with the conversion back to multiple classes

Attributes for learning

GATE supports two types of attribute for relation learning

Argument attributes

- these are attributes that describe an argument, e.g. the part-of-speech or semantic type of an argument, or of its context

Relation attributes

- these are attributes that describe the relation as a whole, rather than a single argument

Attributes may be “windowed” in the same way as for entity learning

Ngrams are also supported, as for entity learning

The following slides give examples of relation attributes

Attributes for learning

Org. Person Organization

In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

Organization Person

Tony Trotter --- BBC

Tony Trotter --- Analysts Inc

Tony Trotter --- Goldman Sachs

Tourre --- BBC

Tourre --- Analysts Inc

Tourre --- Goldman Sachs

Attributes for learning

Org. Person Organization

In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

Organization Person

Tony Trotter --- BBC	distance=2
Tony Trotter --- Analysts Inc	distance=1
Tony Trotter --- Goldman Sachs	distance=5
Tourre --- BBC	distance=14
Tourre --- Analysts Inc	distance=6
Tourre --- Goldman Sachs	distance=3

Attributes for learning

Org. Person Organization

In a BBC interview, Tony Trotter of Analysts Inc said that Goldman Sachs front-man Tourre had his nose in the trough

Organization Person

Tony Trotter --- BBC	distance=2	direction=org-pers
Tony Trotter --- Analysts Inc	distance=1	direction=pers-org
Tony Trotter --- Goldman Sachs	distance=5	direction=pers-org
Tourre --- BBC	distance=14	direction=org-pers
Tourre --- Analysts Inc	distance=6	direction=org-pers
Tourre --- Goldman Sachs	distance=3	direction=org-pers

Creating attributes

- Attributes are created from annotation features
- Except for the most basic of relation attributes, standard GATE PRs will not give you features that are useful for learning relations
- You must therefore create other features that are relevant to your problem
- This will involve writing a custom PR, or some JAPE with Java RHS
- You could combine this with the code that creates instances

Creating attributes

Many useful attributes can be created by combining features from shallow processing PRs

For example,

- Distance between arguments
- Argument order
- Concatenated POS between arguments
- Concatenated token strings between arguments

The Configuration File

Looking at the configuration file

- We set the annotations and features that we want to use as instances, arguments, and class in the Batch Learning PR configuration file
- You will find a configuration file in your hands-on materials, called ***relations-config.xml***
- **Open it using a text editor**

Multiple relation classes

BigBucks Bank hired Tourre, recently fired by Goldman Sachs

Tourre fired-from GS

Tourre hired-by BigBucks Bank

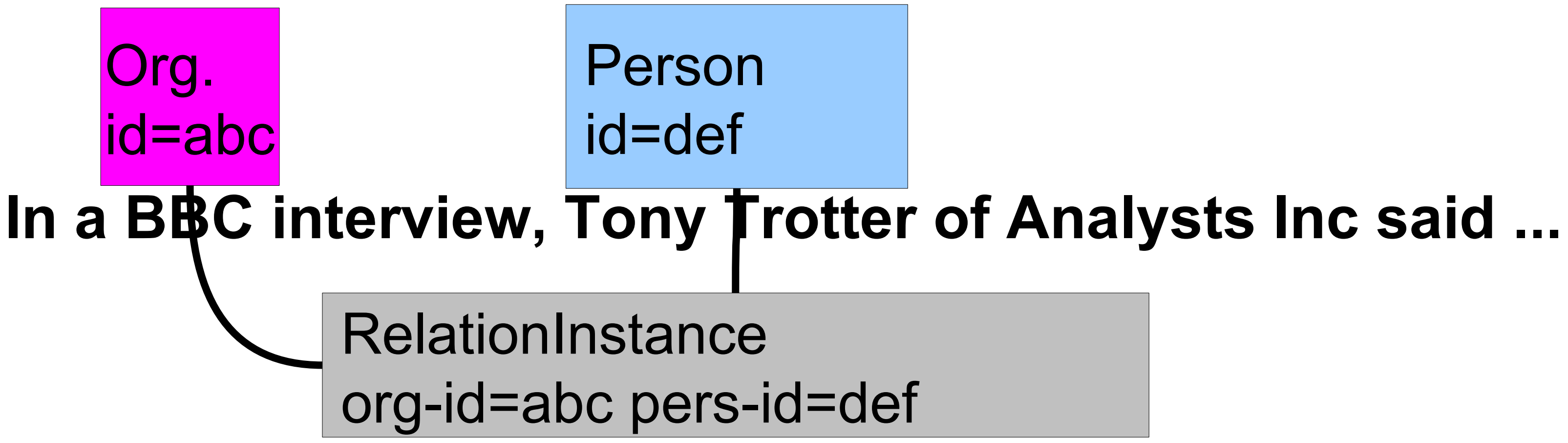
```
<multiClassification2Binary method="one-vs-others" />
```

- We could have several classes (employed-by, fired-from, hired-by etc.). As with entities, we can split this multi-class task into binary classes:
- **one-vs-others**
  - e.g. employed-by vs fired-from + hired-by / fired-from vs employed-by + hired-by / hired-by vs employed-by + fired-from
- **one-vs-another**
  - e.g. employed-by vs fired-from / employed-by vs hired-by / fired-from vs hired-by

•There are clearly performance implications!

•We use one-vs-others for our simple binary example

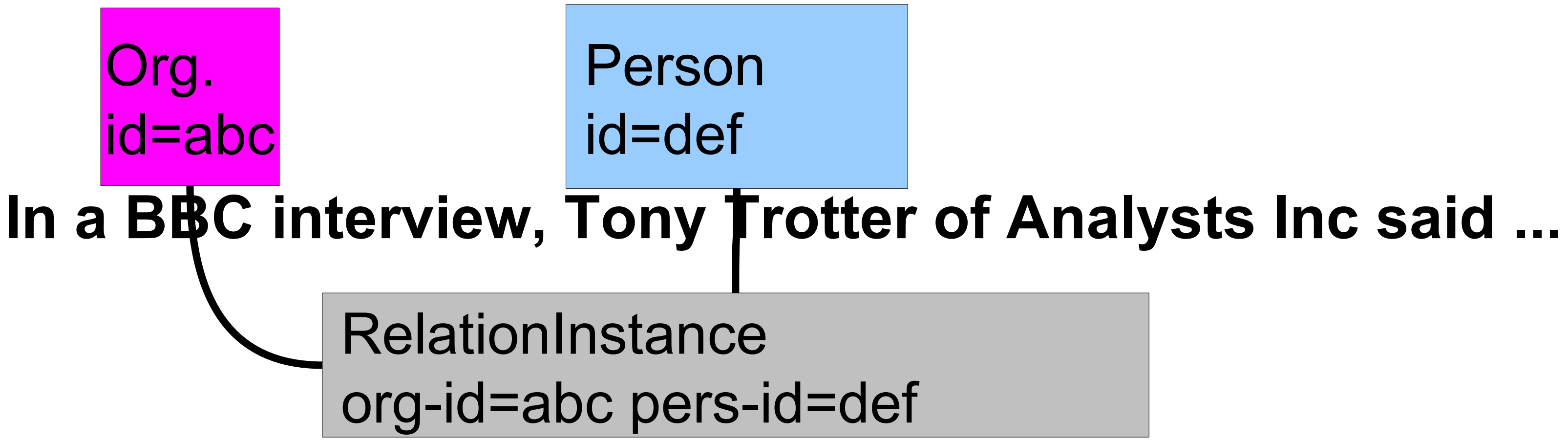
Instances



```
<INSTANCE-TYPE>RelationInstance</INSTANCE-TYPE>
<INSTANCE-ARG1>org-id</INSTANCE-ARG1>
<INSTANCE-ARG2>pers-id</INSTANCE-ARG2>
```

- We tell the ML PR what our relation instance annotation is, and what features point to its arguments

Arguments



```
<FEATURES-ARG1>
    <ARG>
        <NAME>ARG1</NAME>
        <SEMTYPE>NOMINAL</SEMTYPE>
        <TYPE>Organization</TYPE>
        <FEATURE>id</FEATURE>
    </ARG>
    <ATTRIBUTE>...</ATTRIBUTE>
    ...
 </FEATURES-ARG1>
```

Define two arguments, -ARG1 and -ARG2

Define the annotation type and feature of the argument

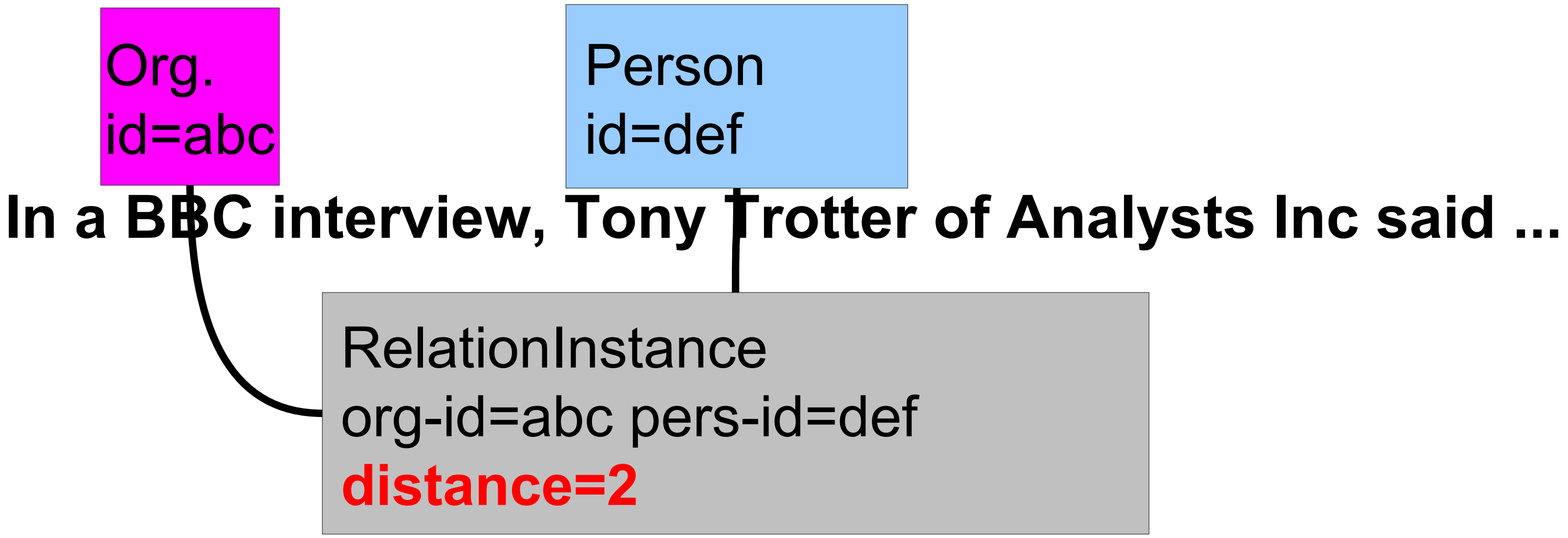
Define attributes in the same way as for entity learning

Argument attributes

```
<ATTRIBUTELIST>
      <NAME>Form</NAME>
      <SEMTYPE>NOMINAL</SEMTYPE>
      <TYPE>Token</TYPE>
      <FEATURE>category</FEATURE>
      <RANGE from="-2" to="2"/>
</ATTRIBUTELIST>
```

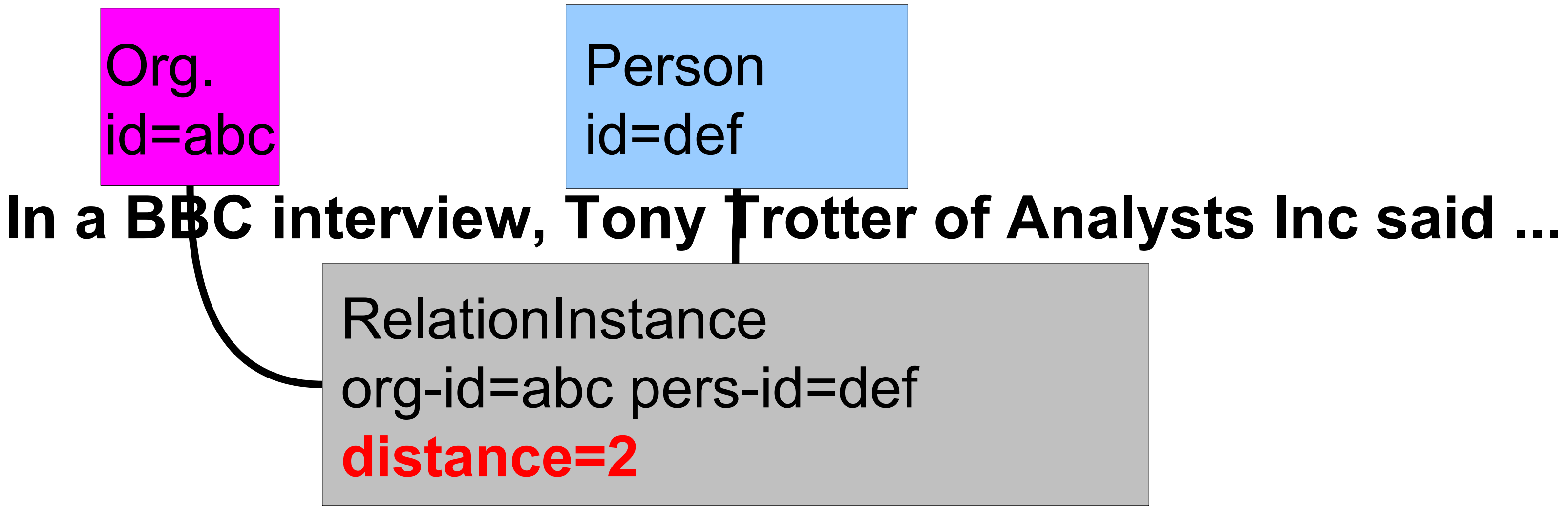
- For argument attributes, we create a specification like the one above, defining the annotations and features that will be used to create attributes of the learning instances
- This is the information from which the PR will learn, so it is important to give it some good data
- You can see in the configuration file that there are several attributes, providing a good range of information
- However, if you have too many attributes it can take a very long time to learn!

Relation attributes



- Relation attributes are defined after the arguments
- They describe an annotation type and feature that contains the attribute, what features relate it to the arguments, and its positional relationship to the instance
- In many cases, it makes sense to use the RelationInstance itself as the source of attributes, as above – but you do not have to
- For example, you could use Token features as attributes

Relation attributes



```
<ATTRIBUTE_REL>
    <NAME>Distance</NAME>
    <SEMTYPE>NOMINAL</SEMTYPE>
    <TYPE>RelationInstance</TYPE>
    <ARG1>org-id</ARG1>
    <ARG2>pers-id</ARG2>
    <FEATURE>distance</FEATURE>
</ATTRIBUTE_REL>
```

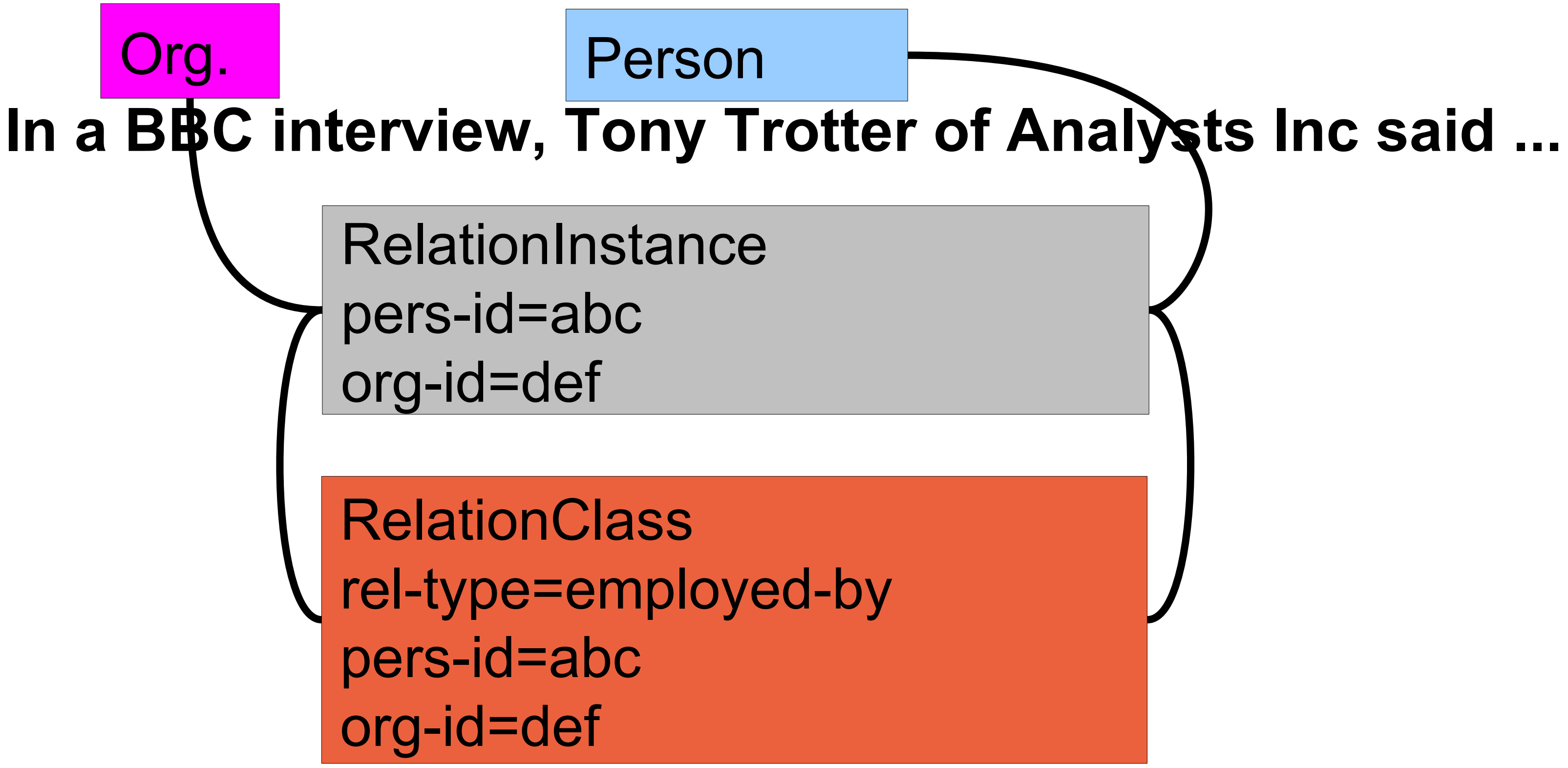
Similar to the ATTRIBUTE and ATTRIBUTELIST elements, but uses ARG1 and ARG2 features to relate to the instance, instead of RANGE or POSITION

Relation class

```
<ATTRIBUTE_REL>
    <NAME>Class</NAME>
    <SEMTYPE>NOMINAL</SEMTYPE>
    <TYPE>RelationClass</TYPE>
    <ARG1>org-id</ARG1>
    <ARG2>pers-id</ARG2>
    <FEATURE>rel-type</FEATURE>
    <CLASS/>
</ATTRIBUTE_REL>
```

- As with entity learning, there must be a single class attribute
- This describes the annotation and feature that give the class of the instance

Relation class



Hands-on exercises: relation learning

Hands-on exercise: introduction

The exercises use a corpus annotated by ANNIE for entities, and by hand with examples of an “enployed-by” relation

It is a toy example, but it does work and shows you how to configure the Batch Learning PR

A full scale application would require more work and coding, developing features specific to the task

We will run the exercise in evaluation mode, looking at the output of the Batch Learning PR's built-in QA

But as with the classification exercises, you could partition the corpus to run a training and test application, and could use the GATE QA tools

RED is things for you to do

BLUE is configuration

The presenter will first run through the basic steps on the next four slides, and then leave you to repeat and follow the exercises on the rest of the slides

Hands-on exercise: create the corpus

Create a new corpus in GATE

Populate it from the directory

- **`module-11-relation-exercise/corpus`**
- non-recursively
- UTF-8

Examine the annotations

The corpus:

- Has 93 documents
- Has been preprocessed with ANNIE to give mentions
- Has 145 manually created RelationClass annotations
- This annotation has a rel-type feature, with a single value, ***employed-by***

Hands-on exercise: make instances

Load the application

- `make-relation-instances.xgapp`

Examine the application

It does some basic pre-processing

Then it runs a JAPE grammar to pair up all Organizations and People annotations into RelationInstance annotations

Run the application and examine the results

Hands-on exercise: examine the config file

We now need to add learning to our application

First, we shall look again at the configuration file for the Batch Learning PR

As before, you will find a configuration file in your hands-on materials, called ***relations-config.xml***

Open it using a text editor

Look at the features and class in the dataset section

Hands-on exercise: first learning

We will now add this learning to our application

Load the Learning creole repository

Create a new Batch Learning PR

For the ***configFileURL*** parameter, browse for the config file we just examined

- `relations-config.xml`

Add the PR to the end of the application pipeline

Set the inputAS and outputAS to ***Key***

Set the mode to ***Evaluation***

Run the application!

Hands-on exercise: first results

- Look at the results
- **Note these down. We will try several different configurations – if you note them down, you will have results to compare**
- Look at the configuration file, and the attributes we used for learning
- The rest of the exercise will look at creating and changing features

Hands-on exercise: making features

First, we will add a JAPE grammar to add some features to our instances

Create a new JAPE Transducer PR

For the grammarURL, browse to the JAPE file make-relation-instances-and-features.jape

In your GATE corpus pipeline, remove the existing make-instances JAPE transducer, and replace with the new transducer that you have just created.

Make sure to set the inputAS and outputAS to Key

For now, also remove the Batch Processing PR from your pipleline

Hands-on exercise: making features

Run your new pipeline

Examine the RelationInstance annotations that it creates, and note the new features

- **poslist** is a list of the POS tags of all tokens between the arguments
- **genposlist** is the same list, but the POS tags have been generalised to the first two characters (so NNP and NNS are both now NN)
- **order** gives the order of the arguments
- **distance** gives the number of tokens between the arguments

(If you understand complex JAPE, you might also like to look at the JAPE grammar that created these features)

Hands-on exercise: configuring features

Now put the Batch Learning PR back at the end of the pipeline. Make sure the inputASName and outputASName are set to Key, and learningMode set to Evaluation

Edit the Batch Learning PR configuration file. Add a section to use the poslist feature as a learning attribute. To help you with this, see the examples in relations-config-extra.xml

Save the file, and re-initialise the Batch Learning PR (right click it and use the menu)

Run the application, and write down the results

More exercises

Try the other features, by adding them to the configuration file. At first, try them one at a time (i.e. remove or comment out the other features)

Each time write down the results

Further exercise ideas:

- _ Try the PAUM algorithm
- _ Try features in combination
- _ Partition your corpus into two corpora of 83 and 10 documents each. Use the 83 documents to train a model, and then apply it to the other 10. When you apply, you will need to set the Batch Learning PR's outputASName to something other than Key
- _ Other feature ideas:
  - Ngrams – see the example in relations-config-ngram.xml
  - Windowing argument features – see the example in relations-config-window.xml