Dan Mangiarelli

Text Mining

GATE is a strong platform upon which to build custom text analysis and information extraction applications. The platform ships with several generic processing resources which must be combined and enhanced to build a custom application. One such application is to use GATE to build a set of tools to extract information from documents related to financial exchanges, such as company loans and acquisitions.

Building a corpus pipeline to recognize such financial transactions involved several steps and several revisions. The current pipeline consists of all of the default ANNIE PRs and some additional custom PRs. The first additional PR created was a Gazetteer list. This relatively small list included only a handful of terms spread among three individual lists. Two of the lists dealt with loaning and borrowing transactions, while the final list contained conjugations of “buy”, “purchase”, and “acquire”. Individual lists were required for “loan” and “borrow” (and their conjugations) because of the roles of the grammatical subjects and objects in each case. Given the phrase “IBM loaned Microsoft”, IBM issued the loan and is the grammatical subject. In the case of “IBM borrowed from Microsoft”, however, IBM is still the grammatical subject, but is the borrower. The use of the passive voice further complicates designing the application by reversing the role of the grammatical subject and object in the transaction. The division of the two lists is necessary because of a series of JAPE rules that are later used to identify the roles of the subject and object, given the form of the verb. The subject and object in the list of purchasing verbs serve the same role, so that did not need to be further divided.

Recognition of verb phrases was important for this text analysis task, so a verb phrase chunker was needed. One is provided as part of the standard GATE distribution, though it must be included as a plugin (the “VP Chunker”, under the “Tools” CREOLE resource). When added to the pipeline, this PR used POS tags to create “VG” annotations over the verb groups it was referring to. While this was largely sufficient, it had some bugs. The underlying implementation is simply a JAPE file, so I made another JAPE file to address one of its bugs. The bug it addresses is that in certain cases, some “VG” annotations are not properly marked as negations (the “neg” feature is not set). This was resolved by a JAPE rule that simply searched for a token with the string “not” within a VG, and updated the VG’s existing features with the proper value for “neg”. It is sufficient for the most obvious deficiency, and is not likely to cause undesired side effects. The other main issue I observed was that phrases in the form of “may not be [verb]ing” (such as with “may not be borrowing”), the verb’s token is improperly POS-tagged, so the VG chunker cannot properly include that as part of the VG annotation (which simply spanned “may not be”). This has not been resolved because of the inaccurate POS tag on the verb. Interestingly, the positive form of that construct is properly annotated.

Noun phrase chunking is also a useful addition to this analysis. However, the built in NP chunker frequently incorrectly annotates, for instance “IBM ten million dollars” from the phrase “Microsoft loaned **IBM ten million dollars**” as a single noun chunk. IBM is part of the same noun chunk as the amount of money, which is incorrect for our purposes. Thus, the NP chunker was not included in the pipeline.

Following the built in VP chunker was a JAPE transducer that ran the custom grammar I wrote, based on the specification that were discussed and developed over the course of the semester. The first phase included the JAPE file to fix the previously mentioned missing “neg” features on VGs. The next phase referenced the Gazetteer list and, combined with information about the verb’s voice, created a FinancialVG annotation with two features: grammatical subject, and grammatical object. For the case of loaning verbs, the values of the features were either “issuer” or “borrower”, while the purchasing verbs were given either “purchaser” or “purchasee” (which is a really poor-sounding word choice, but there is no apparent alternative). The values of the subject and object features referred to the roles played by the subject and object.

The following was designed to be completely generic. It had about twenty rules to match different types of verb phrases. For example, these rules had patters for phrases such as “A verbs B”, “A has *verb*ed B”, “A has been *verb*ing B”, “A will *verb* B”, etc. “A” and “B” refer to the subject and object in the sentence. More specifically, they refer to the parties that issue or receive the money, in the case of loaning verbs. If the item being loaned was present, B specifically referred to the indirect object (e.g. “A loaned $100 *to* ***B***). These rules assigned certain features to a new annotation, FinanicalVerb. These features included such information as the agent’s role, the recipient’s role, the status of the action (presently, progress and commitment).

For example, one of the rules is as follows:

|  |
| --- |
| Rule: A\_has\_verbed\_B  Priority: 50  (  ({FinancialVG contains {VG.voice == "active"},  FinancialVG contains {VG.tense == "PrePer"},  FinancialVG contains {VG.type == "FVG"}}):fvg  ):fv  --> :fv.FinancialVerb = {  agent = :fvg.FinancialVG.grammaticalSubject,  recipient = :fvg.FinancialVG.grammaticalObject,  isAgentSubject = "yes",  actionProgress = "completed",  actionCommitment = "no",  rule = "A\_has\_verbed\_B"  } |

The above rule matches the FinancialVG annotations that contain a VG annotation (from the VP chunker). It looks at VG’s features, including voice, tense, and type. Each combination is part of a separate rule so each case can be handled separately: “has loaned”, “has been loaned”, “should have been loaned”, among several others, are all individual rules. The voice and tense differ in each LHS, and may also match a “neg” feature for negation, and “modal” feature for constructs including “should” and “may”. In this case, we’re looking for a **pre**sent **per**fect verb in the **active** voice. In addition, each RHS is customized for that rule. Two main features carry information about the state of the action: action committed, and action progress. The committed feature refers to whether there is a commitment to complete the transaction. For some cases, there is no commitment. However it is used for positive future cases. It also indicates whether an action is “recommended” (“should”) or “possible” (“may”) in the case of modal verbs. The action’s progress includes information about whether the action has been completed, will be completed in the future, or whether nothing will be completed at all.

Given the sentence “IBM has loaned Microsoft…”, “agent” feature is assigned the value of “issuer”, meaning that IBM serves as the issuer of the loan in that case. The “recipient” is “borrower”, indicating that Microsoft is borrowing the loan. However, the entities are not actually associated with the transaction yet. This is preparation for the next phase which creates that association. The “isAgentSubject” feature is also used in the following phase. It indicates whether the agent in the transaction is also the subject, or whether the recipient is the subject: that is, is the grammatical subject the borrower or the issuer? This is dependent on the voice: when the voice is passive or absent, the roles of the subject and object are different from the roles if the verb was in the active voice.

Another slightly more complicated rule:

|  |
| --- |
| Rule: A\_should\_not\_have\_been\_verbed\_B  Priority: 45  (  ({VG.tense == "HaveVBN",  VG.neg == "yes"})  ({FinancialVG contains {VG.voice == "none"},  FinancialVG contains {VG.tense == "Pas"},  FinancialVG contains {VG.type == "PART"}}):fvg  ):fv  --> :fv.FinancialVerb = {  agent = :fvg.FinancialVG.grammaticalObject,  recipient = :fvg.FinancialVG.grammaticalSubject,  isAgentSubject = "no",  actionProgress = "completed",  actionCommitment = "no",  negative = "yes",  rule = "A\_should\_not\_have\_been\_verbed\_B"  } |

This rule matches a negative construct in the past tense: “IBM should not have been loaned *something* by Microsoft…” In this case, the VP chunker created two annotations: one for “should have been” and another for “loaned”. The former is matched by the “HaveVBN” pattern, while the latter is matched by the “FinancialVG” pattern. The resulting “FinancialVerb” annotation sets “agent” to “issuer”, “recipient” to “borrower”, and “isAgentFirst” to “no”. This indicates that the subject, IBM, is the recipient (the borrower). It is not a modal phrase despite having “should” because the verb in the first VG is “be”, not the financial verb. The FinancialVG has no voice and is in the past tense. The resulting features are similar to the previous example, except that the action has already been completed, even though it has effectively been recommended against.

This phase uses appelt matching. This is because the patterns are of similar length, and we need to properly identify which rules should fire. We also don’t want multiple rules firing on the same area of text. Patterns with more features, such as “modal” and “neg” have higher priorities so they are fired, and not rules with an LHS that doesn’t specify those features. However, because the VGs’ features are specifically matched, that may not have been necessary.

The final JAPE phase looks for organizations or persons surrounding FinancialVerb annotations within the same sentence. Using the agent and recipient features of the FV annotation created in the previous phase, this creates a final Transaction annotation that copies most of the features from the FV annotation. It also copies the values of the FV’s features and applies them to the Transaction, so information about the status of the action is preserved. The agent and recipient features are replaced by the actual roles (e.g. loaner, borrower, purchaser, etc.) and the text of the surrounding entities becomes the value for those features. The resulting features include the name of the role (loaner, borrower, etc.) as the feature name, and the value is the actual name of the entity in that role. That is all of the information we have processed and determined thus far.

Some limitations are that, at this point, the processing pipeline, doesn’t address every aspect of the transaction, like the amount of money or the terms of the loan. It also only works within a single sentence. This restriction is implemented by the inclusion of “Split” on some of the phases, and is to avoid unexpected results in cases that aren’t properly addressed yet (cases where one party may be mentioned in another sentence). It also does not distinguish indirect objects from direct objects, so in some cases, the entity may be incorrectly attributed (e.g. “IBM was loaned $1000 to work with Sun Microsystems by Oracle” would identify “Sun Microsystems” as one of the entities, even though it is part of the terms); rather, it merely looks for entities surrounding the FV annotation. It also should be expandable to other types of transactions beyond the current purchase and loan verbs.