



We use the classifiers and algorithms supplied as part of the Mallet Machine Learning Package [5]. This is in addition to PERL scripts (used for pre- and post-processing) and training, testing, and gold standard data supplied as part of the project.

## II. METHODOLOGY

Our IGT detection system has three labels; IGTB (denoting the beginning or line 1 of an IGT instance), IGTR (remaining lines of an IGT instance or inside an IGT), and NON\_IGT (non IGT lines, or outside an IGT instance).

The feature templates make an important component of the system because learning classification, i.e. what makes a line IGTB or IGTR or NON\_IGT depends primarily on what distinguishing features define the respective label. All our features are binary (1 denoting presence and 0 denoting absence). Barring the last 4 features all are represented as regular expressions. Our final system contained the feature templates as indicated Table 1. The Perl scripts used to extract the features from the text files are named “convert\_data.pl” and “convert\_data\_2.pl.”

FEAT. NAME	FEATURE DESCRIPTION	FEATURE DETECTION
regex1	Presence of number / alphabet denoting beginning of IGT [e.g. “(1)”, “a.”]	<code>/^\s+(\d{1,2}) [a-z]\.?)\s+.+/</code>
regex2	Presence of quote at beginning of a sentence, denoting an IGT translation line	<code>^\s+'/</code>
regex9	Presence of quote at end of a sentence, denoting an IGT translation line	<code>^\s+'/</code>
regex3	Presence of $\geq 8$ spaces at beginning denoting indentation characteristic of an IGT instance	<code>^\s{8}./</code>
regex5	Presence of a gloss line containing a gram – 2 [e.g. “cook-NM-DAT”]	<code>^\w+(\[.\-][A-Z])+\s+/</code>
grams1	Presence of at least 1 gram (denoting an gloss line)	<code>if (numGrams &gt; 0)</code>
grams2	Presence of at least 3 grams (denoting an gloss line)	<code>if (numGrams &gt; 3)</code>
grams3	Presence of at least 5 grams (denoting an gloss line)	<code>if (numGrams &gt; 5)</code>
num_tokens	Presence of atleast 10 words in a line (denoting a non IGT line)	<code>if (numTokens &gt; 10)</code>

**TABLE 1: Final Set of Feature Templates**

Our final system as included in the wrapper IGT\_detect consists of the following modules described in the flowchart below. The yellow boxes indicate input files and the green ones denote output files.

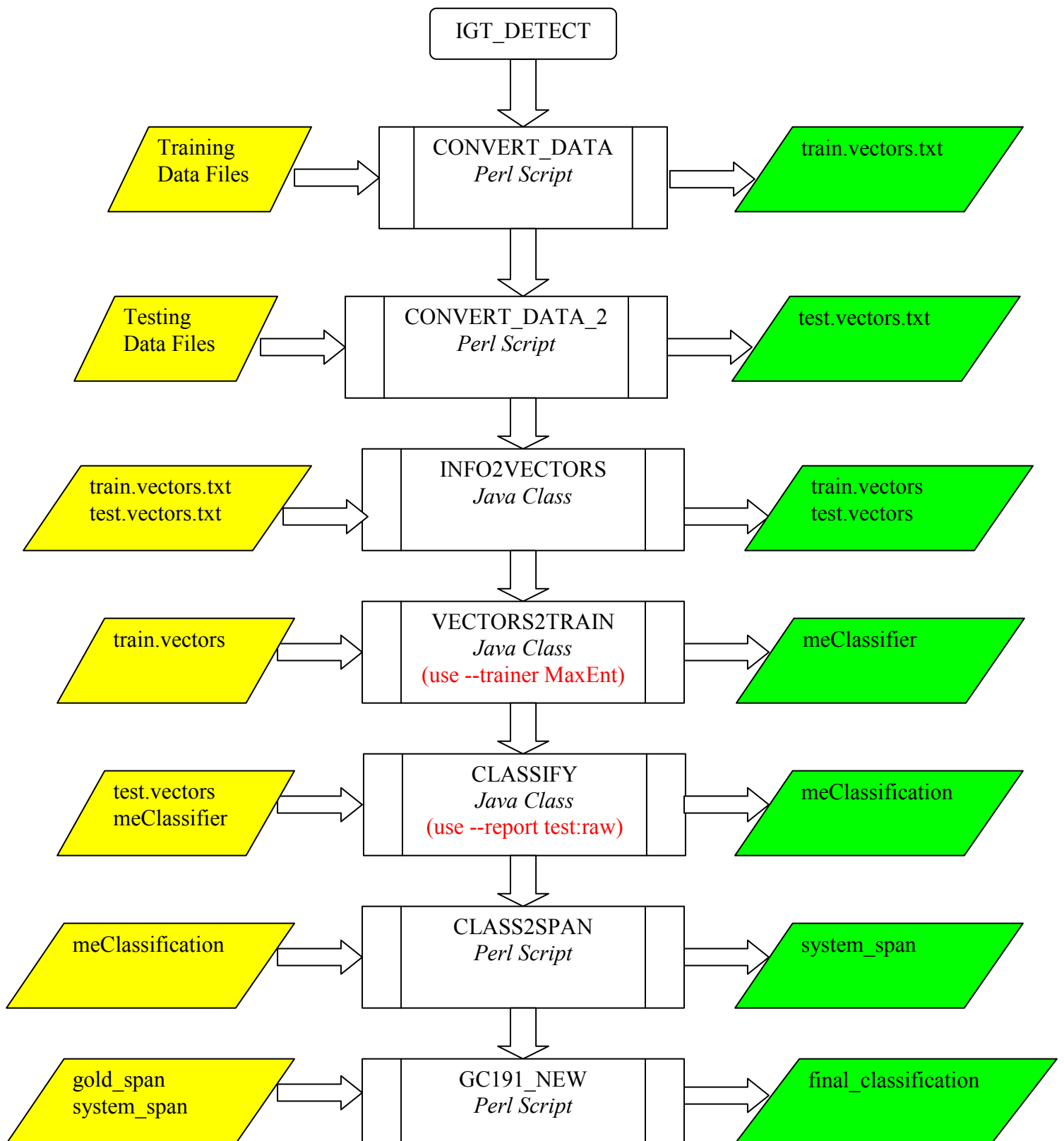


FIG 1: Flowchart of the IGT Detection System

The results of the experiments are described in the next section while the parameters are defined in this section. Our baseline system, i.e. the one with which we started Hw5 had the following features defined as given in Table 2.

FEAT. NAME	FEATURE DESCRIPTION	FEATURE DETECTION
regex1	Presence of number / alphabet denoting beginning of IGT [e.g. “(1)”, “a.”]	<code>/^\s+(\d{1,2})[a-z]\.?)\s+.+/</code>
regex2	Presence of quote at beginning of a sentence, denoting an IGT translation line	<code>/\s+'/</code>
regex3	Presence of $\geq 6$ spaces at beginning denoting indentation characteristic of an IGT instance	<code>/^\s{6}./</code>
regex4	Presence of a gloss line containing a gram - 1 [e.g. “bring-1sg”]	<code>/^.*([0-9a-zA-Z]+-[0-9a-zA-Z]+).*/</code>
regex5	Presence of a gloss line containing a gram - 2 [e.g. “cook-NM-DAT”]	<code>/w+(\[.\.[A-Z]+\)+)\s+/</code>
num_tokens	Presence of atleast 10 words in a line (denoting a non IGT line)	<code>if (numTokens &gt; 10)</code>

**TABLE 2: Initial (Baseline) Set of Feature Templates**

For Hw6, we implemented BeamSearch opting for no pruning and using topN values ranging from 0(default) to 10. All other parameters were kept at default. We failed to achieve any improvement. In fact our system performed much below our baseline numbers as shown in the Experiments section. This makes us believe that there might be a flaw in our algorithm. However owing to time constraints, we were unable to investigate further.

As part of the modifications for Hw7 to help improve our performance we incorporated the following:

1. **Training and Testing:** Using more training data: We trained our baseline classifier on about half the files from \$hw5Dir/more\_training\_data and used MaxEnt Classifier. We used only half the number of total files because we found that training time increased with addition of features.
2. **Data Conversion:** We edited the feature templates from the baseline feature set described in Table 2 to the final system feature set described in Table 1. We added the features grams1, grams2, and grams3. These grams were detected by comparing with an edited (removing certain ambiguous grams) version of the grams file supplied by Dr. Will Lewis. We removed regex4. We changed the value of the number of spaces in regex3 from 6 to 8 spaces. We also added the feature regex9.
3. **Post-processing:** We included some heuristics in the form of a Perl script named “class2span.pl” to induce a better mapping from classification results to IGT spans. These are illustrated in Fig 2. These heuristics implemented after we get

the classification results from Classify.java help improve the number of IGT instances detected; because they take into account the interdependence of the labels. While diagrams A and B in Figure 2 led us to better mapping, diagram C was not successful in span improvement. We discovered that we were unable to detect gloss lines well. This led us to implement the 3 new features grams1, grams2, and grams3 to enable gram detection in gloss lines. Diagram D helps ensure that we don't count just single isolated lines as IGT.

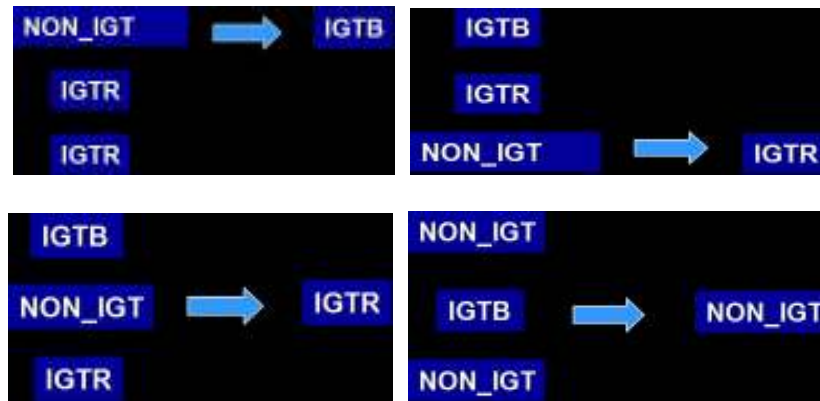


FIG 2: Post processing Heuristics; A(top-left), B(top-right), C(bottom-left), D(bottom-right)

### III. EXPERIMENTS

This section presents our results for the different experiments we performed, as mentioned in the previous Methodology Section.

Our baseline system using the feature templates given in Table 2 gave us the following precision, recall and F-measure numbers using NaiveBayes (Table 3) and MaxEnt (Table 4) classifiers..

	PRECISION	RECALL	F-MEASURE
EXACT	5.24 %	5.82 %	5.51 %
ALMOST	91.13 %	101.12 %	95.86 %

TABLE 3: Naïve Bayes Results on Baseline System

	PRECISION	RECALL	F-MEASURE
EXACT	16.19 %	26.62 %	20.14 %
ALMOST	74.69 %	93.29 %	82.96 %

TABLE 4: MaxEnt Results on Baseline System

Our BeamSearch algorithm (MaxEnt Trainer) performed poorly as evident in Table 5.

	PRECISION	RECALL	F-MEASURE
EXACT	0.57 %	3.80 %	0.99 %
ALMOST	21.90 %	100 %	35.93 %

**TABLE 5: BeamSearch Results on Baseline System**

For experiments involving all the 3 modifications incorporated into our system in Hw7, we got the following results

	PRECISION	RECALL	F-MEASURE
EXACT	53.61 %	38.26 %	44.65 %
ALMOST	81.15 %	59.51 %	68.79 %

**TABLE 6: Hw7 Modification (1, 2, 3) Results on Modified System**

We also explored the performance of our IGT system when we implemented just Modification # 3, i.e. post processing heuristics as demonstrated in Fig. 2.

	PRECISION	RECALL	F-MEASURE
EXACT	62.77 %	39.60 %	48.56 %
ALMOST	97.88 %	59.78 %	73.84 %

**TABLE 7: Hw7 Modification (3) Results on Modified System**

Next we performed experiments by **grouping features** and try to examine the impact of a particular feature group on our IGT detection system. The features mentioned in the Table caption are the ones which were **REMOVED** for that experiment. Thus the numbers reflect what the absence of the specified feature does to the system.

	PRECISION	RECALL	F-MEASURE
EXACT	53.80 %	36.47 %	43.47 %
ALMOST	80.53 %	55.93 %	66.01 %

**TABLE 8: Group 1: Removal of Beginning of IGT, i.e. regex1**

	PRECISION	RECALL	F-MEASURE
EXACT	23.08 %	6.04 %	9.57 %
ALMOST	48.72 %	12.98 %	20.49 %

**TABLE 8: Group 2: Removal of quote marks in Translation line, i.e. regex2 and regex9**

	PRECISION	RECALL	F-MEASURE
EXACT	54.39 %	36.02 %	43.34 %
ALMOST	80.41 %	54.59 %	65.03 %

**TABLE 9: Group 3: Removal of indented lines / leading whitespace, i.e. regex3**

	PRECISION	RECALL	F-MEASURE
EXACT	0.00 %	0.00 %	0.00 %
ALMOST	100.00 %	5.59 %	10.59 %

**TABLE 10: Group 4: Removal of grams in Gloss line, i.e. regex4, regex5, grams1, grams2, and grams3**

	PRECISION	RECALL	F-MEASURE
EXACT	54.39 %	36.02 %	43.34 %
ALMOST	80.41 %	54.59 %	65.03 %

**TABLE 11: Group 5: Removal of word count in line, i.e. num\_tokens**

Note, we were unable to supply the classification results as part of our experiments. This is because we forced the testing file to have NON\_IGT tags by default. Hence we could not get the correct numbers using the output given by Classify.java.

#### IV. DISCUSSION

Our IGT detection system runs successfully on PONGO and completes the classification and spanning in about 20 seconds. With regards to job division, we agree that both of us contributed equally to the project and learnt together the pitfalls of sequence labeling using a third-party code.

We used Mallet classes and algorithms for both IGT detection (Project 2) and document classification (Project 1) tasks. Both of us consider this a learning experience. We were able to explore algorithmic implementation of many machine learning techniques and know more than what we started with. Nevertheless we do have a few nitpicks. Some parts of the code were unclean, containing comments like, “this does not work” or “have to implement this.” We also think the pipes need a more extensive documentation with perhaps more examples of implementation. One point in favor of Mallet though is it covers a lot of ground.

There were some things which stumped us, things we thought would work but did not. Chief among these is the Beam Search. If time permitted, we would have investigated problems with our Beam Search algorithm and also experiment with a larger range of feature templates. We were also thinking of using more tags, at least one more and try to examine its impact on the overall system performance.

With respect to lessons learned, we think that group work and collaboration (including discussion with other teams) are beneficial to tackling such large problems. The group presentations helped us quite a bit in understanding the different parameters and their influence on the general system. Also, it’s hard to know what the right balance is in such a large system wherein a multitude of factors affect the system performance. If we try to improve one fault, it detracts the performance from another angle. We also feel that classification algorithms do not translate well to the sequence labeling task.

## V. CONCLUSION

Our final system, i.e. our best performance was achieved implementing only the post processing heuristics as shown in the re-copied Table 7. The complete commands are supplied in the read-me along with the tarred copy of our complete system.

	PRECISION	RECALL	F-MEASURE
EXACT	62.77 %	39.60 %	48.56 %
ALMOST	97.88 %	59.78 %	73.84 %

**TABLE 7: Hw7 Modification (3) Results on Modified System**

After examining the results of grouping features, we conclude that gram detection, i.e. identification of gloss lines in an IGT instance is the most important feature of our IGT detection system. Therefore, when we remove the gram detection features from the system, the performance decrements to 0 % for exact matches and approximately 10 % for almost matches.

Therefore, we conclude that in the machine learning approach to IGT detection, gram identification of gloss lines and mapping the span files to better output are the two most important components for optimum performance.

## VI. REFERENCES

1. Natural Language Processing Blog > “Getting Started In: Sequence Labeling,” November 1, 2006 entry. <http://nlpers.blogspot.com/2006/11/getting-started-in-sequence-labeling.html>
2. L. Ramshaw, and M. Marcus. 1995. Text Chunking using Transformation-Based Learning. In *Proceedings of the Third Workshop on Very Large Corpora*, Cambridge, June 1995, pp. 82-94
3. Lecture Notes, Ling 572 > “Sequence Labeling and Beam Search,” February 15, 2007, Fei Xia. Slides at: [http://courses.washington.edu/ling572/winter07/teaching\\_slides/seqLabel\\_beamSearch.ppt](http://courses.washington.edu/ling572/winter07/teaching_slides/seqLabel_beamSearch.ppt)
4. ODIN: The Online Database of Interlinear Text, Website: <http://www.csufresno.edu/odin/>
5. MALLET: Advanced Machine Learning for Language, Website: <http://mallet.cs.umass.edu>