4. Character Segmentation, Language Models and Beam Search

The heart of Tesseract

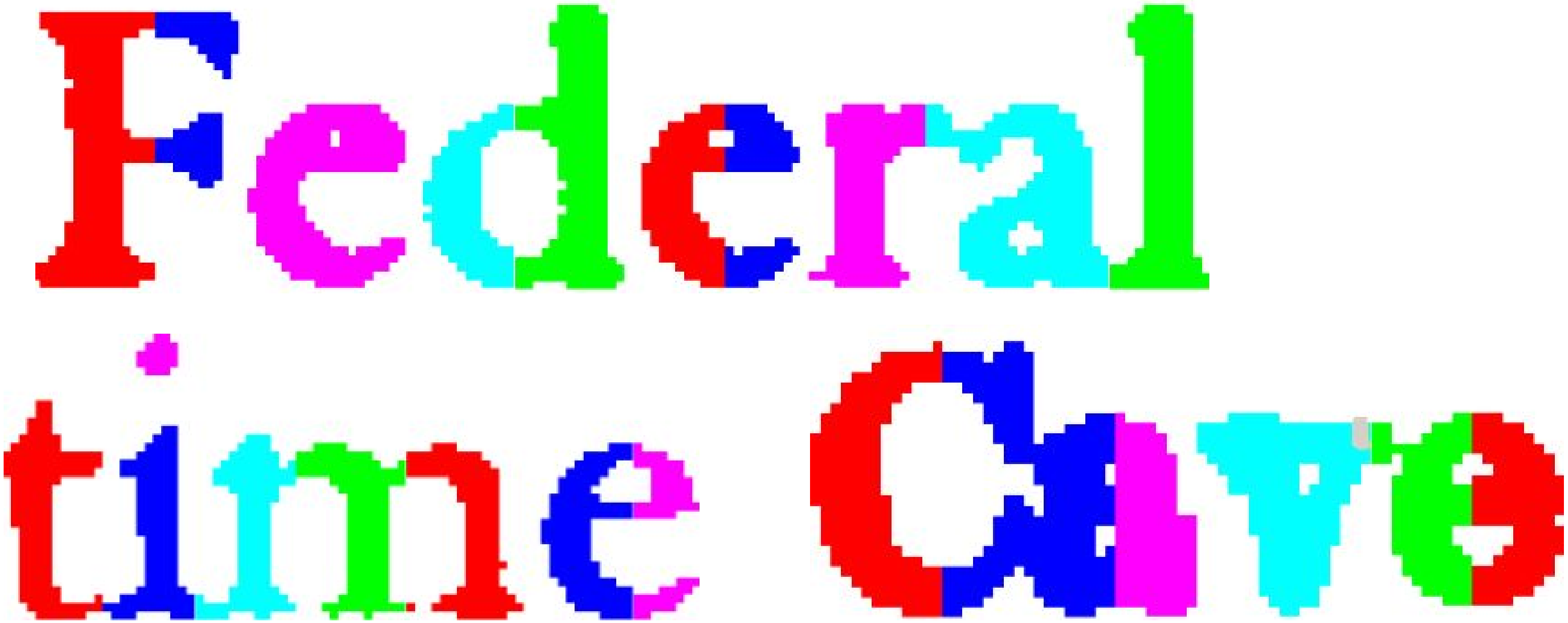
Ray Smith, Google Inc.

Approaches to Segmentation

- Segment first using only geometry.
- Maximally chop, then combine with a beam search. (Over-segmentation.)
- Sliding window to "avoid" segmentation altogether.
- Tesseract: Chop only as needed, then combine as needed.

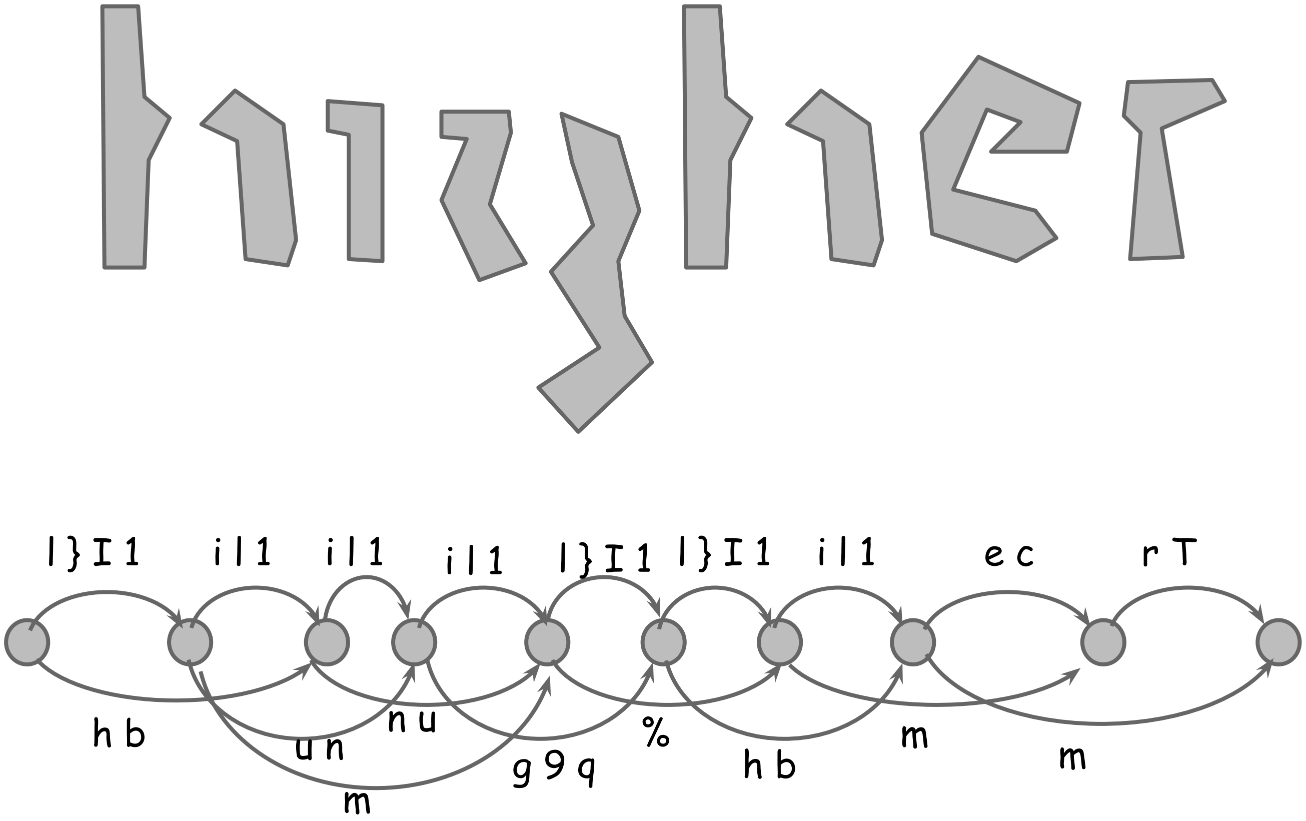
Over-Segmentation

- Aim is to maximize recall of chops with the compromise of reduction in precision.

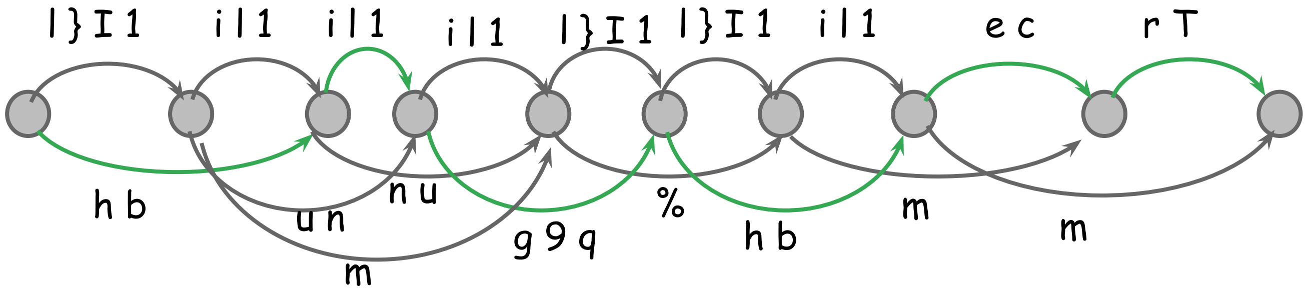


Segmentation Graph

- Segmentation possibilities and classifier results form a directed graph

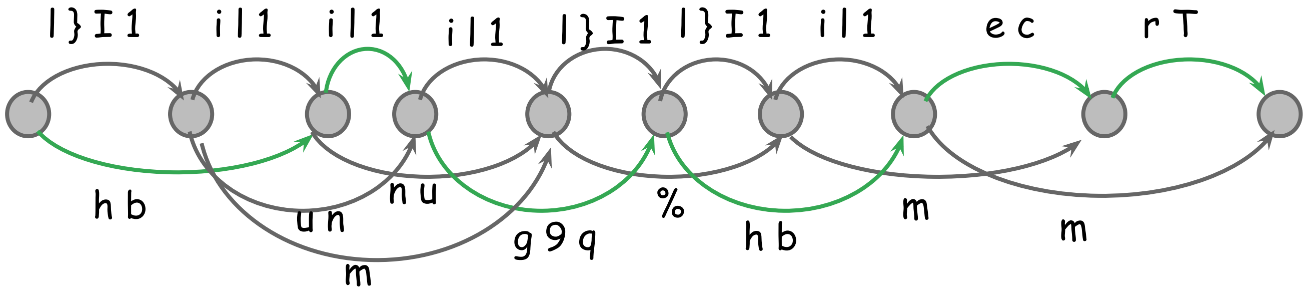


Searching the Segmentation Graph

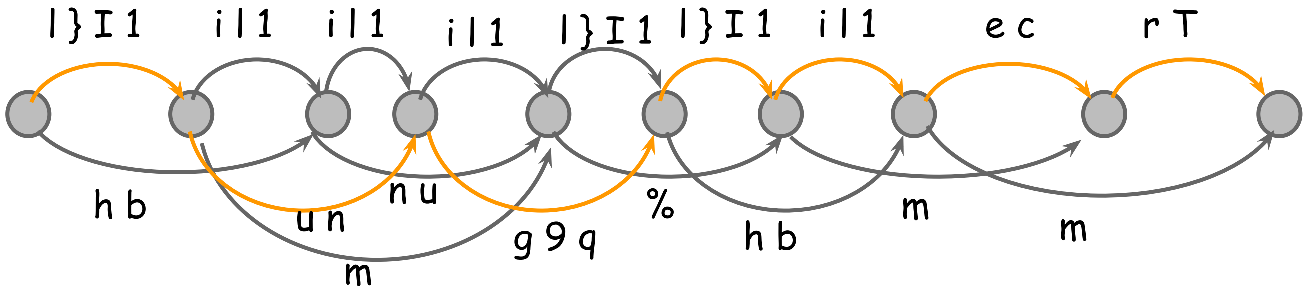


higher

Searching the Segmentation Graph



higher



}uglier

Integration of Language Models (General Methods)

- Implement Language Model as Finite State Machine.
- Search Language Model and Segmentation Graph in parallel.
- Combine “probabilities” in some sensible way.
- Hidden Markov Model methods are good example.

Segmentation Free = Extreme Over-Segmentation

- Slide over the word/textline with a classifier/HMM.
- Beam search + shape model probs + language model probs solves the segmentation internally.
- Really just an extreme form of over-segmentation.

Tesseract Segmentation Approach based on observations:

- Initial segmentation is often correct or close.
- Classifier generally doesn’t like incorrectly segmented text.
- Over-segmentation often leads to poor results., eg m->iii

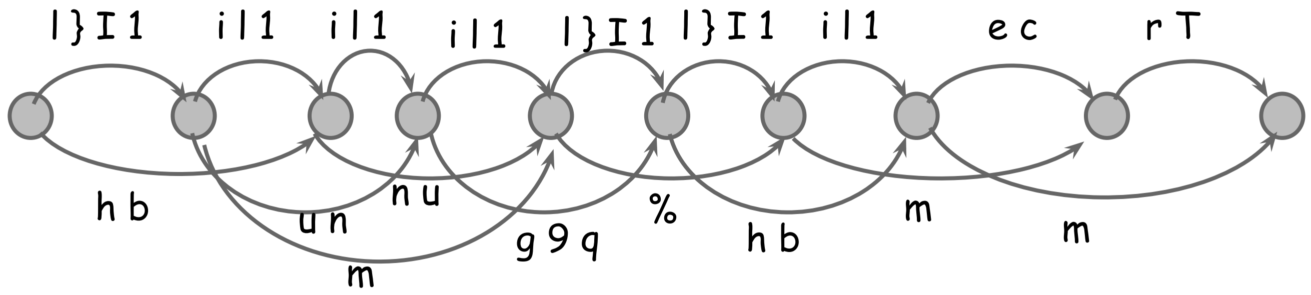
Tesseract Segmentation Approach

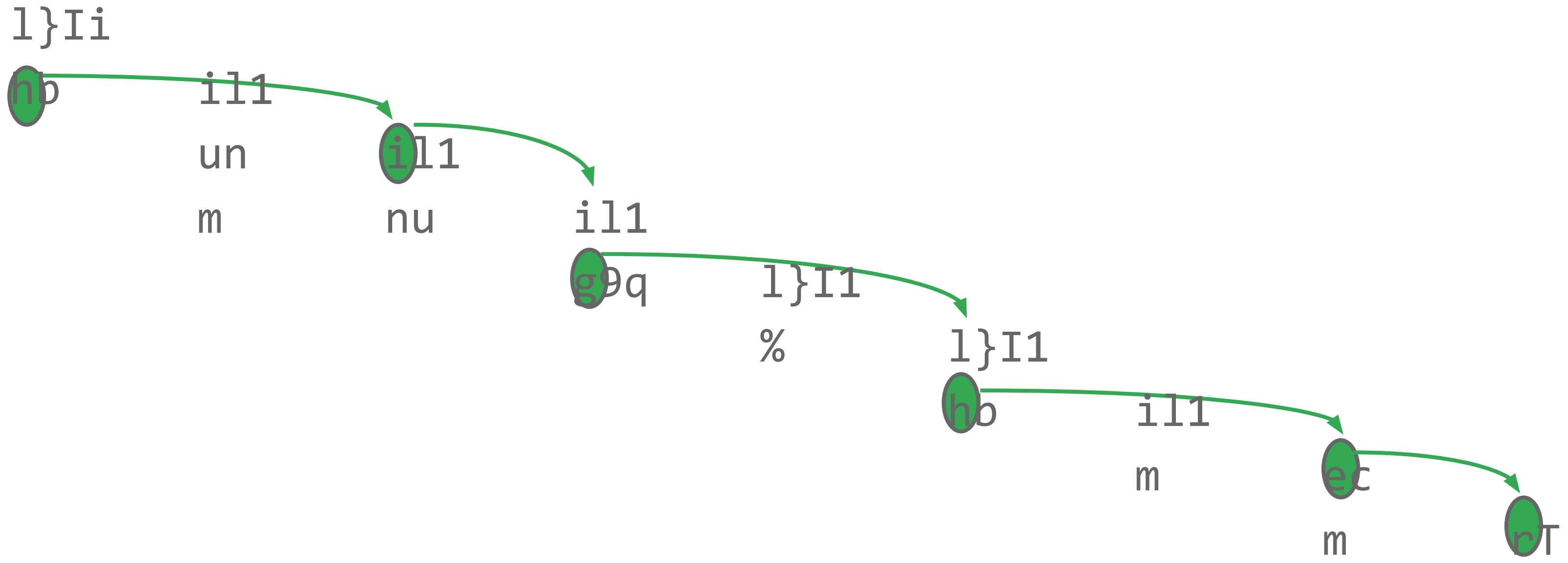
```
Classify Initial Segmentation
Search Word: OK?  Yes => Done
while any Bad Blob has any Chops available
    Chop and classify pieces of Worst Choppable Blob
    Search Word: OK?  Yes => Done
while any fixable “Pain point”
    Associate adjacent blobs and classify
    Search Word: OK?  Yes => Done
```

Types of Pain Point

- Initial: Join each adjacent pair
- Ambiguity: Eg m/rn
- Path: Neighbors of blobs in the current best path

Ratings MATRIX = Segmentation Graph





Each entry holds a BLOB_CHOICE_LIST providing classifier choices with rating and certainty.

Evaluation of a WORD_CHOICE (no params-model)

$$\text{Word Rating} = \text{word\_factor} \sum_{\text{segmentation}} \texttt{blob\_choice->rating()}$$

word_factor =

Condition	base word_factor	Add-ons
Frequent dawg word	1.0	Inconsistent case +0.1
Other dawg word	1.1	Inconsistent case +0.1
Non-dawg word	1.25 +0.01 for each char over 3.	Inconsistent case +0.1 Inconsistent punc +0.2 Inconsistent chartype +0.3 Inconsistent script +0.5 Inconsisted char spacing +0.01 All except script +0.01 for each additional occurrence.

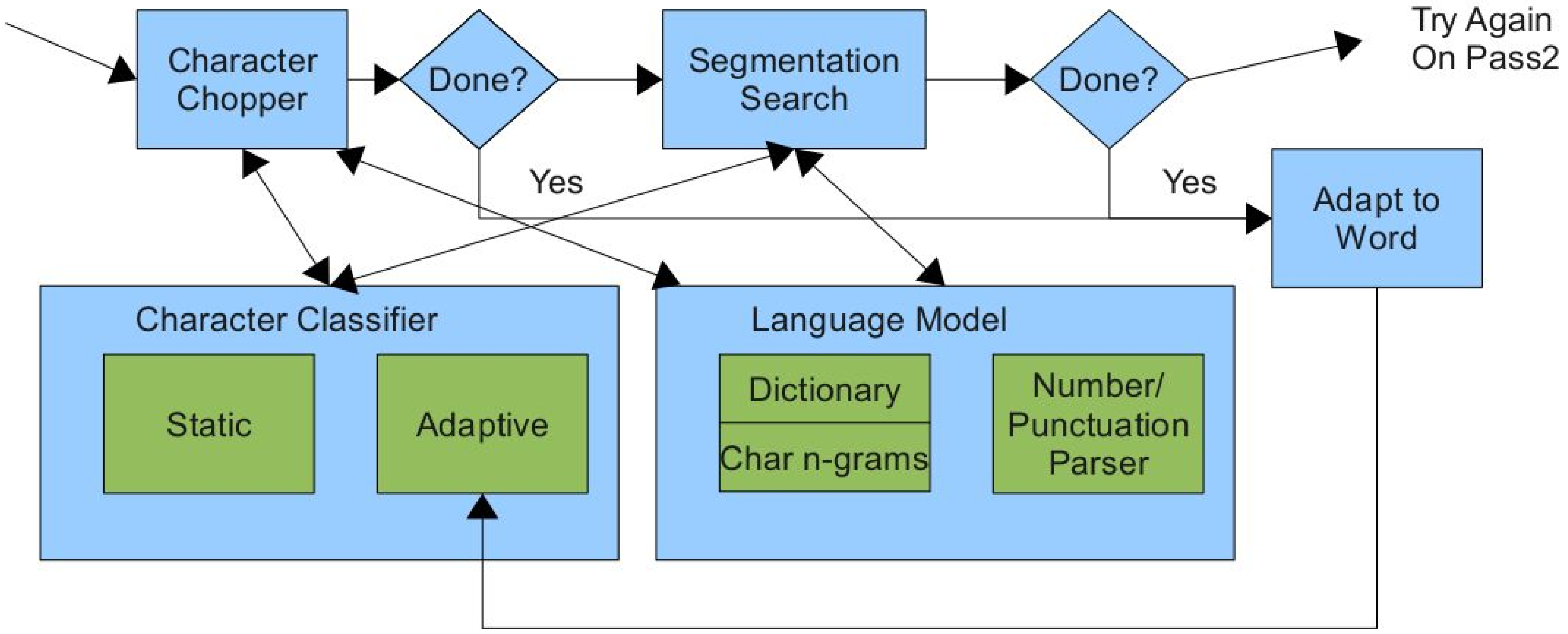
Evaluation of a WORD_CHOICE (with params-model)

$$\text{Word Rating} = \text{word\_factor} \sum \texttt{outline length}$$

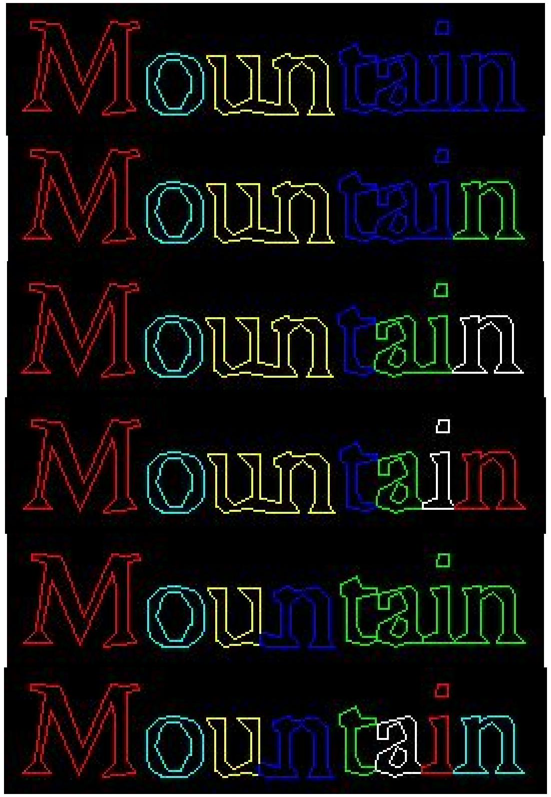
word_factor = weighted sum of word features:

- mean blob rating
- num inconsistent spaces
- num inconsistent char type
- num x-height inconsistencies
- num case inconsistencies
- word length (in type categories)

Tesseract Word Recognizer

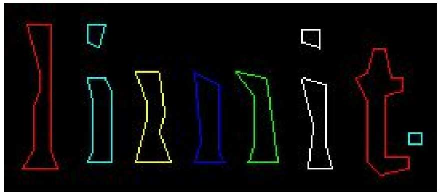
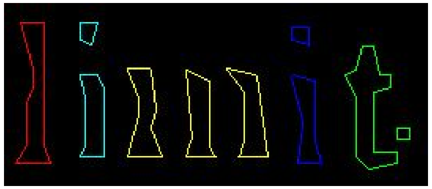
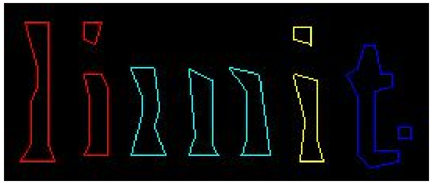
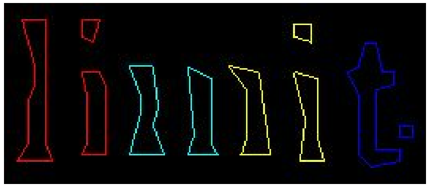
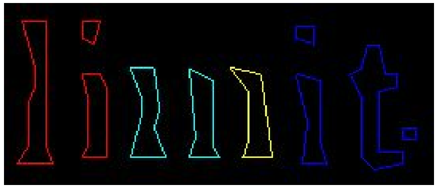


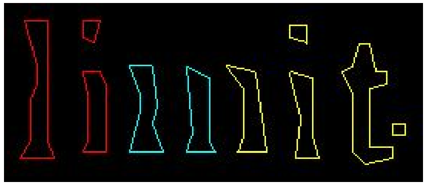
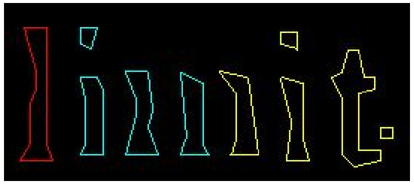
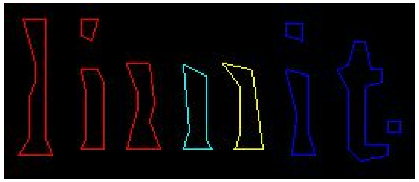
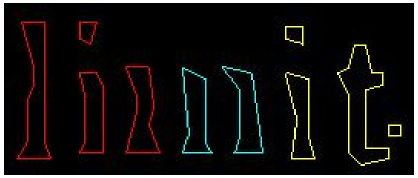
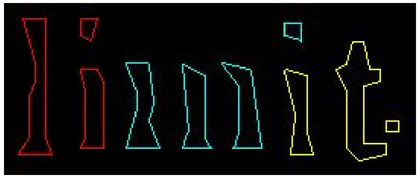
Example of Chopping (unlv/mag.3B/2/8022_028.3B.tif Col 2, line 6, word 1)



Word	Distance	Worst blob	
Momm	212.2	7.7	
Mommn	186.3	8.3	
Momtfln	178.0	9.2	
Momtain	124.9	5.3	
Mounm	184.0	7.7	
Mountain	80.6	3.1	ACCEPT!

Example of Combining (unlv/doe3.3B/4/2214_007.3B.tif, col 2, line 8, word 2)

	Word	Distance
	lilllit-	77.58
	limit	57.1
	Emit	89.7
	Unfit	95.4
	Hulk	122.7
	Bulk	136.8

	Word	Distance
	HUM	120.6
	MUM	127.0
	BUM	134.7
	1mm	112.8
	Milk	147.2
	huh	137.1
	fink	140.7
	Emu	129.7
	BMW	140.3

Thanks for Listening!

Questions?