1. The accuracy on test partition is: 89.2673% . I used most frequent tag for unknown words.
The sentence level accuracy is low: 15% (just checked for curiosity)
To run:
To train: python problem-1.py train (train.tagged file in the same directory)
To test: python problem-1.py test (test.tagged file in the same directory)

The trained model file is saved to a file save_model.txt included.

2. When did a add 1 smoothing:
Accuracy is: 87.08485% (replace unknown words with most frequent tag)
Accuracy is: 77.028263% (unknown words ignored)
Using the same model trained from problem1.
To run:
python problem-1.py smooth

3. python problem-3.py
Has some issues, its just giving 25.4%accuracy. Just included

4. to train: python nlp_hw_pos_rnn.py --datapath `pwd` --train temp.model --num_epochs 1 --embedding_approach pretrained
To test: python nlp_hw_pos_rnn.py --datapath `pwd` --test pos.nn.5.model --embedding_approach pretrained
Remember to provide the --embedding_approach=pretrained during test when trained on a pre-trained word embedding
Uses and repurposes much of Michael Capizzi's code of the tutorial on Dynet
Model is too huge unable to attach.
without pretrained vectors:
overall accuracy: 89.32685512367491%   //(5epocs) 39.86 - 1epoc , 63.16 - 2epoc, 82.47- 3epoc
with pretrained 87.6%

5. The paper title is part of speech tagging from 97% to 100%. Here he discusses the limitations of the datasets used, the challenges encountered while trying to go beyond 97% and possible options or solutions. He talks about the evaluation measures, firstly getting some tags right is easy, like punction marks but some tokens are ambiguous too. If we consider the sentence level accuracy, accuracy measure where we count points only when we get all the tags in the sentence right, it is perhaps low. It also makes more sense to consider them as making a single mistake, can make the tagger useless for downstream tasks. It should also be noted that when two taggers are asked to annotate, interannotator disagreement rate is high, which means different tags for the same text! It also depends on who is tagging them, his competence, attention to detail, etc. He mentions two foundational issues: Are the labels discrete enough to be assigned as single label to a word? , Is our evaluation measure good enough?. Some incremental improvements discussed are tuning features and parameters - lowering the support threshold for including rare word features, considering both sequences from left and right (eg- 5wShapesDS + distributional similarity).
    Splitting tags was one option considered. You may imagine having more tags, better expresses the feature, but it has been proved to be a waste of time. A thorough error analysis also directs us to some pitfalls. The lexical gap , its hard to capture context, since there are so many, between train and test. Handling unknown words , relying on features that are ambiguous makes it hard. Keeping more history helps but is not a very practical solution. The gold standard itself has wrong tags and is inconsistent. Difficult linguistics needs more information to be captured.
    Some options for correcting the treebank itself were explored, given its inconsistencies. But an attitude originated in IBM largely is largely against it. It comes from a school that considers real world data to be inherently noisy and we need to use it as is, rather than massaging the data to our needs! Whereas many of the inconsitencies in data are systematic and can be fixed by writing some rules. When matching a tree pattern using Tregex(section 5), some issues can be fixed easily: past tense vs past participle, plurals as singulars, 'that' word which is ambiguous for taggers!When couple of 100s of such rules are applied, the accuracy is improved - even for sentense level and unknown words.