

Assignment 3

CMSC 473/673 — Introduction to Natural Language Processing

Due Wednesday October 20th, 11:59 PM

Item	Summary
Assigned	Wednesday October 6, 2021
Due	Wednesday October 20th
Topic	Classification
Points	100

In this assignment you will understand feature design and gain experience in both implementing and using maximum entropy (log-linear) models.

You are to *complete* this assignment on your own: that is, the code and writeup you submit must be entirely your own. However, you may discuss the assignment at a high level with other students or on the discussion board. Note at the top of your assignment who you discussed this with or what resources you used (beyond course staff, any course materials, or public Discord discussions).

The following table gives the overall point breakdown for this assignment.

Question	1	2	3	4
Points	20	10	30	40

What To Turn In Turn in a writeup in PDF format that answer the questions; turn in all requested code necessary to replicate your results. Be sure to include specific instructions on how to build (compile) your code. Answers to the following questions should be long-form. Provide any necessary analyses and discussion of your results.

How To Submit Submit the assignment on the submission site:

<https://www.csee.umbc.edu/courses/undergraduate/473/f21/submit>.

Be sure to select “Assignment 3.”

Full Questions

1. (20 points) Each of the following scenarios describes the result of some (made up) classifier: there is a list of the correct labels, and the corresponding predictions. For each of the following situations, compute accuracy, recall, precision, and F_1 score. You must show work, or some intermediate steps, to receive full credit on this problem.
 - (a) A binary classification result, where the correct labels are [T, T, F, T, F, T, F, T] and the predicted labels are [T, F, T, T, F, F, F, T]. Assume T means “true” (the desired class) and F (“false”) is the “default” class. Compute accuracy, and compute recall, precision, and F1 for T.
 - (b) A binary classification result, where the correct labels are [T, F, F, F, F, F, F, T] and the predicted labels are [F, T, F, F, F, F, F, F]. Assume T means “true” (the desired class) and F (“false”) is the “default” class. Compute accuracy, and compute recall, precision, and F1 for T.
 - (c) A multiclass classification result, where the correct labels are [T, F, U, F, F, F, U, T] and the predicted labels are [F, T, U, F, F, F, F, T]. Assume T means “true,” U means “maybe,” and F (“false”) is the “default” class. Compute accuracy. Then, compute, at both the micro and macro levels, average recall, precision, and F1. Explain why you should *not* include the F class in the precision, recall, and F1 computations.
 - (d) A multiclass classification result, where the correct labels are [C, C, A, C, C, C, C, C, B, A, C, C, C] and the predicted labels are [C, C, C, C, C, C, C, C, B, A, A, C, C]. In this example, there is no “good” default option, so we can consider A, B, and C to be all possible classes/labels of interest. Compute accuracy. Then, compute, at both the micro and macro levels, average recall, precision, and F1. Explain why you *should* include all classes in the precision, recall, and F1 computations.
2. (10 points) Later in this assignment, you will be performing maxent-based classification. The files you will use are derived from (some of) the Universal Dependencies data¹. To make the classification more straight-forward, we’ve done a bit of reformatting of the training, development, and testing files. This question is designed to get you familiar with this data, before you use it for classification.

The data are available as JSON files in the directory

`/afs/umbc.edu/users/f/e/ferraro/pub/473-f21/data/UD-Frames/`.

Each file is a dictionary; the keys are instance IDs,² and the values are the actual labeled sentences (the sentence and a label). Each label is a string, and each sentence is a list of the annotated words (each word is in the Universal Dependencies CoNLL-U format). Notice this is a multi-class classification problem.

Each label is a **semantic frame**. Roughly, a semantic frame is an abstract label for some high-level action or concept. For example, a high-level TRANSACTION semantic frame could encompass the acts of “buying,” “selling,” “negotiating,” and “paying” (among others).

Pick a language to answer the following questions for.

- (a) How many training, development and testing instances are there? (Remember, each entry in the top-level JSON dictionary is an instance.)

¹<https://github.com/System-T/UniversalPropositions>

² The instance ID isn’t actually needed for the task; it’s provided simply to give you an easy way to refer to posts *if* you need to.

- (b) Within each split, some of the sentences are repeated. How many of the sentences are repeated (alternatively, how many sentences are unique)?
- (c) Across all of training, development and testing for your language, how many different labels are there?
- (d) Create a training histogram for the number of labels (e.g., an (unnormalized) prior $p(\text{label})$).
3. **(30 points)** This question guides you through some examples of feature design. **For each of the following scenarios, provide an example feature (or feature template).** For each scenario, you must provide a description in English (prose), and either with math (e.g., as shown in class/on the board, or in Appendix A) or in sufficient pseudo-code. See Appendix A for an extended example.

This question talks about “linguistic annotations” $z_{i,j}$ for a particular word token $w_{i,j}$. Each linguistic annotation $z_{i,j}$ could be, e.g., the word’s part-of-speech tag, or possibly a collection of linguistic properties. As an example of this, in the UD data files, the part-of-speech tags are in the third and fourth columns, while the linguistic properties are in the sixth column. For example, the following (Spanish) UD word $w_{i,4}$ has two types of parts-of-speech tags (columns 4 & 5), morphological features (column 6), and syntactic features (columns 7-9). You could consider any combination of these columns to form $z_{i,j}$. That is, $z_{i,j}$ does not have to be a single label; it could itself have a structure, e.g., $z_{i,4,\text{POS}_1} = \text{VERB}$ and $z_{i,4,\text{MOOD}} = \text{Cnd}$.

	WORD	LEMMA/FORM	POS	POS	MORPH. FEATURES	SYNTAX	MISC.
4	pediría	pedir	VERB	VERB	Mood=Cnd Number=Sing Person=3 VerbForm=Fin	0 root Y	-

Given sentences $s_i = w_{i1}w_{i2} \dots w_{iM}$, their labels l_i and linguistic annotations $z_i = (z_{i,j})_j$ for each word $w_{i,j}$, define features for a discriminately trained classifier $p(l \mid s_i, z_i) \propto \exp(\theta^T f(s_i, z_i, l))$.

- (a) Define a feature that uses only the WORD.
- (b) Define a feature that uses both the WORD and the LEMMA/FORM values.
- (c) Define a feature that uses both the WORD and the POS values.
- (d) Define a feature that uses both the WORD and the MORPH. FEATURES values. (Remember that the value in MORPH. FEATURES effectively defines a dictionary. As a hint, split the MORPH. FEATURES value on the vertical pipe (“|”) and define features based on each key=value item.
- (e) Define a feature that uses the WORD value and multiple of either the LEMMA/FORM, POS, or MORPH. FEATURES.
4. **(40 points)** In this question, you will use maximum entropy models in order to perform sentence-level semantic frame identification. You may use an external library (like Python’s sklearn, Pytorch, Tensorflow, liblinear, etc.) to help you compute, train, and evaluate the model. You may follow any official tutorial for that toolkit; just be sure to cite it. These tutorials may refer to maxent as “logistic regression.” (An official tutorial will be hosted on that toolkit’s website.) You may **not** use unofficial tutorials, such as random blog posts, Stack Overflow, etc.
- **Your Task** Your task is to perform well-controlled examinations of the following models (described below) on sentence-level semantic frame prediction. Specifically,
 - Document internal progress and development by training on the training set and evaluating on the development set.
 - This internal development **must** include some amount of tuning of hyperparameters or model configurations (including what features are defined and any regularization).

- When internal development is done, you are to pick the best configuration for each model class.
- Once that is done, evaluate these best-in-class models on the testing set.
- Present the results, both internal and final, in readable formats, e.g., in tables or plots.
- Identify the 20 most “extreme” features, e.g., those whose weights θ_k have large absolute value, and discuss what associations those features are identifying.
- Discuss, in one to three paragraphs, the overall outcomes of your experiments: what features or model configurations yielded the highest performance? Speculate on why you think this might be.
- Be sure to describe, and justify, any design decisions or assumptions you made along the way.

Turn in all code, including what is necessary for producing the plots. Clearly indicate in the writeup what model and model configurations resulted in the best test set performance.

- **How You Will Be Graded** Your grade on this problem will be from proper documentation of your code, experiments, and reporting of results on both the development and test portions.
- **System Specifics**

Models Implement the following per-sentence classifiers, which identify a relevant semantic frame for a sentence:

- (i) A baseline of your choice. The simplest baseline to implement will be “the most common baseline:” identify the label that occurs most often in the training set, and then predict that label for every single test instance. However, you may define your own baseline, if you want to.
- (ii) A maximum a posteriori classifier that somehow utilizes maximum entropy models to model the posterior $p(l | y, z)$, where y are the words of the sentence, l is the sentence’s label, and z are any additional annotations. You can do this by modeling a MAP classifier $p(l|y, z) \propto \exp(\phi^T g(l, z, y))$, directly modeled with a conditional log-linear distribution.

Data Pick a language from the data introduced in question 2. It can be the same language you analyzed in that question, or a different one. You only need to experiment on one language.

Features Experiment on dev data with features that are beyond lexical. This means you should have features beyond those that *solely* look at sequences of words. You can draw inspiration from the features you designed in the previous question, from Appendix A, or from discussion either with classmates or from Discord. (Just remember to credit any ideas.)

As a reminder, the previous question had you think about features for “linguistic annotations” $z_{i,j}$ for a particular word token $w_{i,j}$. Looking at the same Spanish UD word $w_{i,4}$ as before:

	WORD	LEMMA/FORM	POS	POS	MORPH. FEATURES	SYNTAX	MISC.
4	pediría	pedir	VERB	VERB	Mood=Cnd Number=Sing Person=3 VerbForm=Fin	0 root Y	-

your beyond-lexical features could consider how the parts-of-speech are used (in conjunction with the words, or not); the individual morphological features; and/or how syntactic features impact semantic frame prediction.

Implementation For the implementation, you may either use a pre-existing maxent implementation (toolkit), or you may implement your own. For example, you could code one up in Pytorch, Tensorflow, Keras (with the Tensorflow backend), SciPy, liblinear, or another toolkit.

Predicting Prediction should be done on a per-instance basis: for each key-value pair in the JSON files, make a prediction.

Evaluation Evaluate your predictions using precision, recall, and F_1 . While you may implement these yourself, you may also use 3rd party libraries. For instance, the Python library `sklearn` (available through the Anaconda build) has functions `sklearn.metrics.precision_score` and `sklearn.metrics.recall_score`.

Appendix A Feature Design

You can do a whole lot more feature design:

- Your features can look at certain properties of a word, such as if the first letter is capitalized, e.g.,

$$f_{\text{is-capitalized}}(w) = \begin{cases} 1 & w \text{ is capitalized} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

For ease, I have not included any sort of direct label association in this feature definition, though you would definitely need to do so. For this feature, $f_{\text{is-capitalized}}(\text{cat}) = 0$ while $f_{\text{is-capitalized}}(\text{Cat}) = 1$. Notice that this feature does not depend on the vocabulary at all: it simply fires if the first character is an upper-case character. Feature like this, that look at how a word/text is written—but not necessarily what that text is—could be helpful for identifying certain types of entities (e.g., company names and other proper nouns are capitalized), or other syntactic tasks (e.g., part-of-speech tagging, especially for German).

- You can use either/both of word embeddings and character embeddings. There are some available for you to use in [/afs/umbc.edu/users/f/e/ferraro/pub/473-f21/embeddings/](https://afs.umbc.edu/users/f/e/ferraro/pub/473-f21/embeddings/).

How would you use these embeddings? One simple way is to create a feature for every coordinate position. If you're using 300 dimensional embeddings and L different labels, you would have $300 * L$ features: one per coordinate and label combination. For example, the feature for coordinate 121 and label l_1 could simply return the 121st coordinate of a word embedding vector, provided the label under consideration matches l_1 .

While templating can get you greater specificity, it may overparametrize the model; this can lead to poor performance. The issue of overparametrization is generally an empirical one.