Active learning on PoS annotation

Bachelor project – François Delafontaine

***Abstract** – We use the OFROM+ dataset and a CRF model for PoS annotation to experiment with active learning by comparing it to a random data selection. We tried multiple query strategies – including oracle – as well as two models and variations on the data itself and concluded that, for our volume of data, a random strategy seems optimal; we also observed that the volume of available data influences the learning curves.*

Introduction

The OFROM+ corpus (Avanzi et al. 2012) contains transcribed recordings of spoken French that are annotated in PoS (Part-of-Speech, meaning grammatical labels such as verb, noun, etc.) using an automatic annotation tool, DisMo (Christodoulides et al. 2018). That tool's accuracy is at 0.97-98, yet the annotation remains too faulty for research. This motivates better metrics to evaluate an annotation tool, as well as better reference data for the purpose of training models. Active learning focuses on the latter.

Active learning is an iterative process that seeks to maximize the learning curve, that is, to get the highest accuracy with as little data as possible. Our goal will be to compare passive (traditional) and active learning for PoS annotation with OFROM+ data, with the hypothesis that active learning will generate a better learning curve: our model will learn faster with it.

We will present the data and model (1), then present the process of our experiment and its factors (2) before discussing the results (3).

1. Data & Model

Our data comes from a database of spoken French counting ~2 million datapoints[1]. We will present that data's specifics (1.1) as well as how we pre-processed it (1.2) to turn it into an input. We will then discuss how we partition that data (1.3) and the model we use for our experiment (1.4).

1.1. Data

The OFROM+ corpus is a dataset (from OFROM and CFPR projects) of audio recordings that have been manually transcribed in TextGrid annotation files (Boersma & Weenink 2025). We will, to describe our data, use a series of terms:

• *token*:	a minimal unit corresponding roughly to a word.
• *label*:	the PoS (Part-of-Speech) label attached to a token.

[1] Specifically from the May 2025 version of the OFROM+ corpus.

- *interval*: a set of tokens bounded by two timestamps.
- *sequence*: a set of tokens corresponding to an IPU (Inter-Pausal Unit).
- *tier*: a set of intervals along the audio signal.
- *speaker*: a set of tiers (usually one transcription + annotations).
- *file*: a set of tiers (from all speakers).

Some of those terms describe the TextGrid structure: text is put in *intervals* to align it on the audio using timestamps. *Tiers* are a convenient way to put parallel information along the audio signal: annotations, overlapping speakers, etc. *Tiers* can contain any type of information but, for spoken discourse, we expect a set of speakers and, for each speaker, one reference tier (the transcription) with others attached to it (annotations). A TextGrid file is the set of all tiers related to an audio recording.

Files can be considered as a set of datapoints and the minimal unit for correction and sharing.

The *token* is a technical unit. Text is split (tokenized) using a set of symbols (space, dash, etc.) as boundaries; the resulting list are tokens. Linguistically, the minimal unit would be either the *phoneme* or *morpheme* but in practice, *tokens* have been the minimal unit that oral linguistics has worked with.

As for the *sequence*, it is for the model (see 1.4) and corresponds to an IPU (Inter-Pausal Unit), that is any set of tokens between two pauses. Types of pauses are defined by their duration: there is a threshold under which a pause isn't an IPU boundary anymore. By a *sequence* we mean any arbitrary set of tokens between two pauses of at least 0.5 second[2] (or at the start/end of a *tier*).

Figure 1 showcases a TextGrid with a 5 seconds audio excerpt and three tiers containing, in order: the tokens, their corresponding PoS labels and their corresponding lemmas.

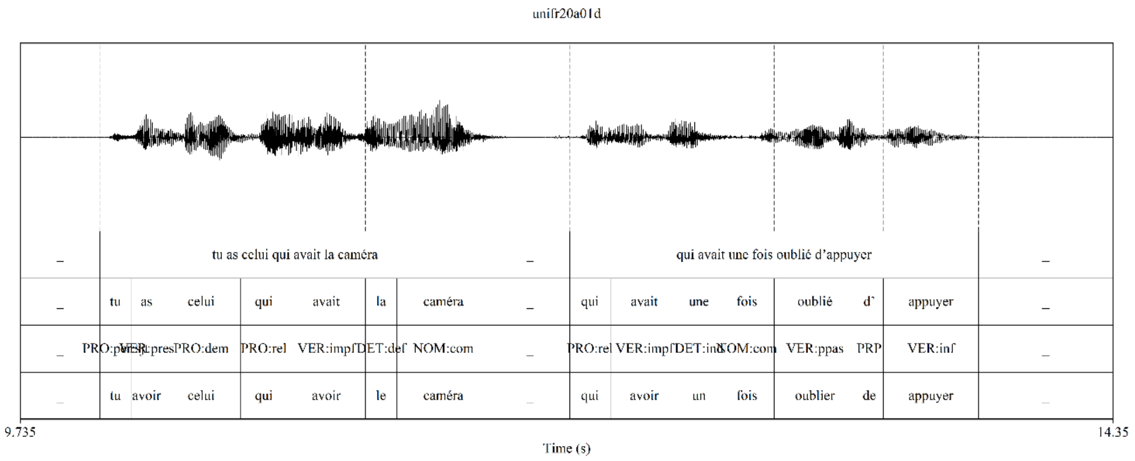
unifr20a01d
_
tu as celui qui avait la caméra
_
qui avait une fois oublié d'appuyer
_
tu as celui qui avait la caméra _ qui avait une fois oublié d' appuyer _
PRO:per VER:pres PRO:dem PRO:rel VER:impf DET:def NOM:com _ PRO:rel VER:impf DET:ind NOM:com VER:ppas PRP VER:inf _
tu avoir celui qui avoir le caméra qui avoir un fois oublier de appuyer
9.735
14.35
Time (s)

Fig.1: OFROM+ transcription

Not all tokens are text. The OFROM+ corpus reserves some symbols for special cases: (a) the "_" pause, (b) the "#" anonymization, (c) the "@" third-party speaker, (d) the "%"

[2] We won't discuss the theory behind that threshold. It can be considered arbitrary.

inaudible speech and (e) the "-" truncation. What matters here is not hat those symbols are or how they are used but that they exist; they do not require further discussion.

Finally, we want to discuss PoS labels. Those will be the classes of our dependent variable in the model. We won't provide an exhaustive list (there are +60 of them) but present their structure. Those labels have fields separated by ":" symbols:

XXX:field:field

The first field is a three-letters code for the main grammatical category: VER for verb, PRO for pronoun, etc. From there, as many fields as needed can be appended, although a single one is the norm and it rarely exceeds two: VER:inf for an infinitive, PRO:per:sjt for a personal pronoun in a subject role, etc. It would be possible to only keep the main category and as a result reduce the number of labels to 12: we choose to maintain all fields, even verb auxiliaries[3].

OFROM+ labels have not been manually set. They are the result of an automatic annotation by the DisMo annotation tool. While it has been partially corrected for this project, it still contains thousands of erroneous labels. Those automatically generated labels will be used to evaluate our model: all accuracy scores should be taken with caution as a result.

1.2. Pre-processing

We have transformed the OFROM+ data prior to our experiment. Most of the transformations are purely technical: for example we separated *tokens* and *labels* and formatted the *tokens* to reduce the computations during the experiment. The important transformations are:

a) We removed all reserved symbols (see 1.1).
b) We grouped the *tokens* in *sequences* and the *sequences* into *files*.

Reserved symbols can be discarded. Each of them can have only one *label* without any ambiguity, meaning we do not need a model to annotate them. Removing them does create gaps in the *sequences*: for example a anonymized name would be removed, leaving its determiner without a noun. Such gaps are close enough to interruptions / revisions, phenomena in spoken discourse, that we expect the model to be minimally impacted. The remaining count of datapoints after removal is presented in figure 3.

Nb. Files	Nb. Sequences	Nb. Tokens
1,475	145,514	2,032,274

Fig.3: Count of files, sequences and tokens.

The pre-processed data has the following structure:

```
File:
- X:
```

[3] Those, we suspect, would be detected at a later stage through linguistic rules.

```
--  sequence:
---    features
- y:
--  sequence:
---    label
```

Meaning each *file*, an object in memory (not on the hard drive), will contain two lists for the independent and dependent variables of our model respectively. Each list in turn contains an equal number of *sequences* that are themselves a list of either *features* or *labels*. One *feature* is the *token* itself but we can add more features to describe it (see 1.4). Code-wise, the features are in a dictionary with 'token', for example, being a key.

1.3. Data partition

Our experiment will use the available data for different purposes. We will use different terms to refer to them:

- *data*: anything from the whole list of *files* to a single *datapoint*.
- *datapoint*: a list of *features* (including the *token*) plus the corresponding *label*.
- *dataset*: all *files*. The entirety of the OFROM+ database.
- *subset*: any set of *files* within the *dataset*.
- *reference dataset*: a *subset* set apart for evaluation.
- *available data*: the dataset minus all subsets.

More specifically, we will use *subset* to refer to the *training set*, meaning the data used to train the model; and we will use *reference dataset* to refer to the *evaluation set*, meaning the data used to evaluate the model. The *available data* then is:

available_data = (dataset – reference_dataset – subset)

This corresponds to any file still not assigned, making them available.

Our *reference dataset* is by no means a gold standard. Our *labels*, being automatically generated, are faulty and can't provide an accurate evaluation. This is only to imitate real conditions where such a manually-annotated, separate dataset would exist. Such datasets usually contain 100k (100,000) *tokens*. Therefore, our *reference dataset* will always have that size.

Due to the nature of that *reference dataset*, nothing prevents us from using either fixed or random *reference dataset* and *subset*: any part of the *dataset* should be equally valid for the experiment and if it is not then replicating the experiment on another part should prevent any bias. We will discuss this further when presenting the experiment (point 2).

1.4. Model

For a PoS annotation, a model of reference is Conditional Random Fields (CRF). A CRF allows for the selection of a *label* based on both the *features* and the position in a *sequence*. Their general principle is (Sutton & McCallum 2010: 23):

$$p(y|x) = \left(\frac{1}{Z(x)}\right) * \prod_{t} \exp\left(\sum_{k} \theta_k f_k(y_t, y_{t-1}, x_t)\right)$$

Starting from a naïve Bayes classifier, it adds a sequential aspect akin to a Hidden Markov Model, as well as a conditional aspect through a logistic regression. Both are contained in the 'f(...)' function via the y_{t-1} and x_t arguments, where 'y' is the label at position 't' and 'x' is the features; we will call 'f(...)' a *feature function*, a function that only considers the *feature* x_k among all *features* 'x' at position 't'; all other *features* are set to 0. As a result, ignoring the auto-regression (y_{t-1}), the sum of those functions (and the exponential) is a logistic regression with θ a coefficient for each *feature*. The auto-regression is the Hidden Markov Model, where the *label* is determined by the *label* at the previous position in the *sequence*: based on the local Markov property, y_{t-1} contains all the positional information needed. The product then is simply the product of the probabilities at all positions.

We implement our CRF through scikit-learn's *sklearn_crfsuite* library. The input is two lists of *sequences* 'X' and 'y' containing the *features* and *labels* respectively (see 1.2).

When referring to our model we will mean a CRF with only a single *feature*: the *token* itself:

1. *token*: the token itself

We may refer to it as our *base model* our *simple model* by opposition to our *complex model* or our model *with added features*. In that case, we will mean that we added a fixed set of features alongside the *token*:

2. *l3*: the last 4 characters of the token
3. *l4*: the last 4 characters of the token
4. *pos*: the PoS if available
5. *p1-3*: the tokens at position -1 to -3 relative to our token
6. *s1*-3: the tokens at position 1 to 3 relative to our token

We will not discuss those features, their justifications, the noise they generate or how they are retrieved. We only want to say that we have two available models; we will only refer to the first one, with a single feature, unless explicitly state otherwise.

The model's hyperparameters ('c1' and 'c2') have been optimized on 100k of our dataset for both the base and complex models: (0.0127, 0.023) and (0.231, 0.0004) respectively.

2. Experiment

Our experiment consists of using our data and model to build learning curves. To do so, we train a model at different *subset* sizes: the training data grows by a fixed amount, which in turn should cause the model's accuracy to grow as well. To generate a learning curve, we follow three steps:

1. Select an initial subset (plus a reference dataset).

2. Train on the subset and evaluate on the reference dataset.
3. Select and add data to the subset, then repeat (2).

Steps (2-3) form a loop that we repeat as many times as we want: we choose 10. As pseudo-code, the experiment would be:

```
X, y = empty_lists
reference_dataset = select_files(…, nb_tokens)
for 10 loops:
    X, y = select_files(…, X, y, nb_tokens)
    crf_model = train(X, y)
    accuracy = predict(crf_model, reference_dataset)
    yield accuracy
```

The *initial subset* is our *subset*, a set of *files* corresponding to a fixed amount of *tokens*. The *reference dataset* is a separate set of *files* corresponding to 100k *tokens*.

Any time we select data, what we select is a *file*, not *tokens*. We select as many files as is needed to reach the fixed amount of *tokens* between two points in our curve. A *file* is the minimal unit for correction and sharing: in real conditions, we could not take just a part of its data. Since files tend to have ~1.3k tokens on average, any amount of tokens under 5k is unreliable. To go under that amount requires reducing or abandoning the file unit: one way to do that is to have a single sequence per file, with as many files as sequences in our dataset. If we further constrain the size of a *sequence* to 5 or less, we can have amounts of data as small as 10 *tokens*.

At each loop we add *files* worth the same fixed amount of *tokens* to the *subset*, re-train and re-evaluate our model to obtain an accuracy score. The core of the experiment is how we select the files to add to the *subset*: the *query strategy*.

The main strategies are:

- *random*: the "passive" learning, files are selected at random
- *confidence*: files are selected through an average confidence score
- *diversity*: there is a redundancy score added to the confidence score
- *oracle*: the confidence score is replaced by the accuracy score
- *token*: the confidence score is limited to a set of tokens

Of all of them, the main "active" strategy is *confidence*: for any file we predict the *labels* with our trained model, then take the confidence score for each *label* and average them. The assumption is that a low confidence score will indicate a case where the model can improve: the lower the score the more useful for training. Since we always pick the next file to add by the highest weight, the strategy's formula becomes:

file_weight = (1 – avg_confidence_score)

The *diversity* strategy adds (1 – redundancy_score) to it, with the redundancy score obtained by comparing (through a cosine similarity) the token distributions of our subset and the considered file. The *token* strategy selects the K lowest-confidence tokens from the subset and only takes their average confidence scores in the file. The *oracle* strategy accesses information we should not have (the *labels*) to give the file an accuracy score: the logic is the same as with the confidence score, the lower the better for training.

Beyond those strategies, there are several parameters for our experiment:

- *size*: the fixed amount of tokens added at each loop
- *fixed*: whether the *subset / reference dataset* are fixed or random
- *features*: whether our model has added features
- *avg*: the method to average the confidence score
- *data_size*: the amount of available data we initially want

The *size* is the size of our *initial subset*, as well as the size of the additional data at each loop. Once the *initial subset* and *reference dataset* have been selected, they are set and cannot change (the *subset* will expand, of course); but between two experiments, they can be different; *fixed* ensures that they stay the same. *Features* determines whether we keep our base model or if we add (a fixed set of) features to it.

We have considered three methods to average the confidence score: *average* for a classic average; *macro* for a macro-average, meaning an average per *label*; *entropy* to measure how varied the confidence scores are. Without any explicit indication otherwise, we use the classic average.

Finally the *data size* is how much data, in *tokens*, should remain once we substract our *initial subset* and *reference dataset* from the *dataset*, before step (3). By default this is all the remaining data in the *dataset*. If we did restrain the amount of data we will specify by how much.

In short, we have one dataset with three variants (size, fixed, data size), a model with two variants (features) and four active strategies with variants (avg). We have not tested and will not discuss all combinations (point 3), nor is the above even exhaustive. For example, we also tried applying a *file cost*, roughly comparable to an ENUA (Expected Number of User Actions, Arora & Agarval 2007: 1), by approximating the number of *tokens* expected to be corrected.

The main experiment is to compare the passive (random) strategy with an active (confidence) strategy. Anything beyond that is designed to expand on the result.

One last remark is that a random selection requires replicating the experiment several times to stabilize the learning curve. For our active strategies, when the subset and reference dataset are fixed, the learning curve will always be the same; when they are random, the learning curve will require several replications as well.

3. Results

Our general hypothesis is that active learning, through a query strategy, will generate a better learning curve: the model will get more accurate with less data. Based on this, we expected an average file confidence query strategy to be sufficient.

This proved mistaken. We will present first the impact we observed from varying the data (fixed, data size) (3.1), then expand on this for smaller sizes (size) (3.2) before covering other considerations (features) (3.3).

3.1. The data factor

Figure 4 shows the results for a 10k (10,000) tokens *size*, meaning our *subset* grows from an initial 10k to 100k in ten *loops*. Each time we plotted the learning curves for the passive (random) strategy, the file confidence strategy and the file diversity strategy. The random strategy is our baseline and we expect the other two to be above it.

We plotted four variants, depending on whether the *initial subset* and *reference dataset* are random or fixed and whether the *available data* is limited (low) or not. When limited, we only kept just enough data to complete all loops. With both *fixed* and *low* settings, the initial and final points is identical for all strategies.

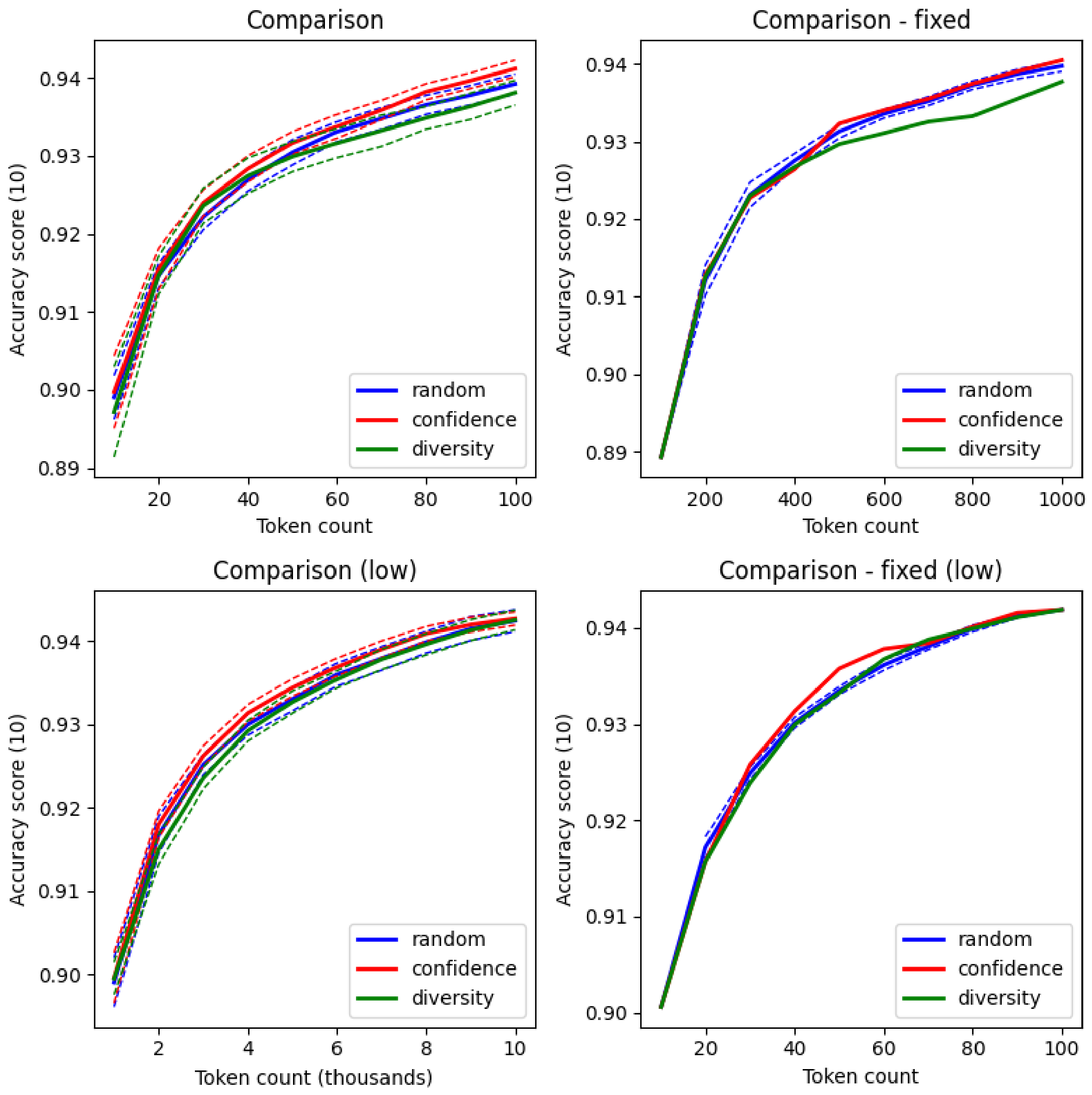

Fig.4: Comparison between random, file confidence and redundancy strategies.

The figure's x-axes provide the *subset* size in tokens; the y-axes the accuracy score with the number of replications (the number of times the experiment was repeated) in parenthesis; the legend indicates the query strategy.

Without discussing the accuracy itself (see 3.3), we can first observe that a *fixed* setting is reliable enough to compare the learning curves. Randomizing *subset* and *reference dataset* adds noise but does not fundamentally alter the curves.

The second observation is that the random strategy, when starting with a *subset* of 10k tokens, is already close to optimal. The diversity strategy either matches it or underperforms while the confidence strategy barely exceeds it in places. There can be multiple causes for this:

a) The model is already fairly trained, with a starting accuracy of ~0.9 from 10k tokens. Active learning may lose in efficacy at this advanced stage.
b) The data distribution may be such that each *file* offers a roughly equal gain to the model. This is contradicted by parts where a strategy differs in efficacy but would explain the general trend.
c) Our data is such that the strategies we consider fail to select the right files.

Chaudhari et al. (2021) warned about the confusion a language dataset can generate: selecting low-confidence *tokens* that may actually just repeat known information. In short, we cannot exclude that our query strategies are at fault.

But our third observation is for us the most important: our strategies perform better, against the random baseline, when the *available data* is limited. This is especially visible on a *fixed* setting but holds true regardless. This, to us, suggests that:

d) The size of the data to select from impacts the efficacy of active learning.

3.2. The size factor

Focusing on the size of our subet (a), on the assumption that our strategies would fare better if the model was in a less advanced state, we repeated the experiment on a *fixed* setting, again varying between a full or limited available data for sizes of 10, 100 and 1k tokens. The results are in figure 5.

The opposite of our expectations happened, with our strategies severely underperforming. Again, the difference between a full and limited *available dataset* is massiv: when all remaining data is available, the learning curve break off completely; only when limited does the diversity strategy beat our baseline and, even then, only for up to 1-2k tokens; the same confidence strategy, in turn, is always slower.

One core difference between those low, 10-to-1k sizes and the 10k experiment is how the dataset is split. In the 10k experiment, an active strategy has to select among about 1.4k files; in the 10-to-1k sizes, the dataset has been split in about 400k files. This again underscores the impact the amount of *available data* has on the experiment. Our next hypothesis would be that by splitting our initial data into files of *size* tokens (the amount needed for a *loop*), we would obtain yet another learning curve.

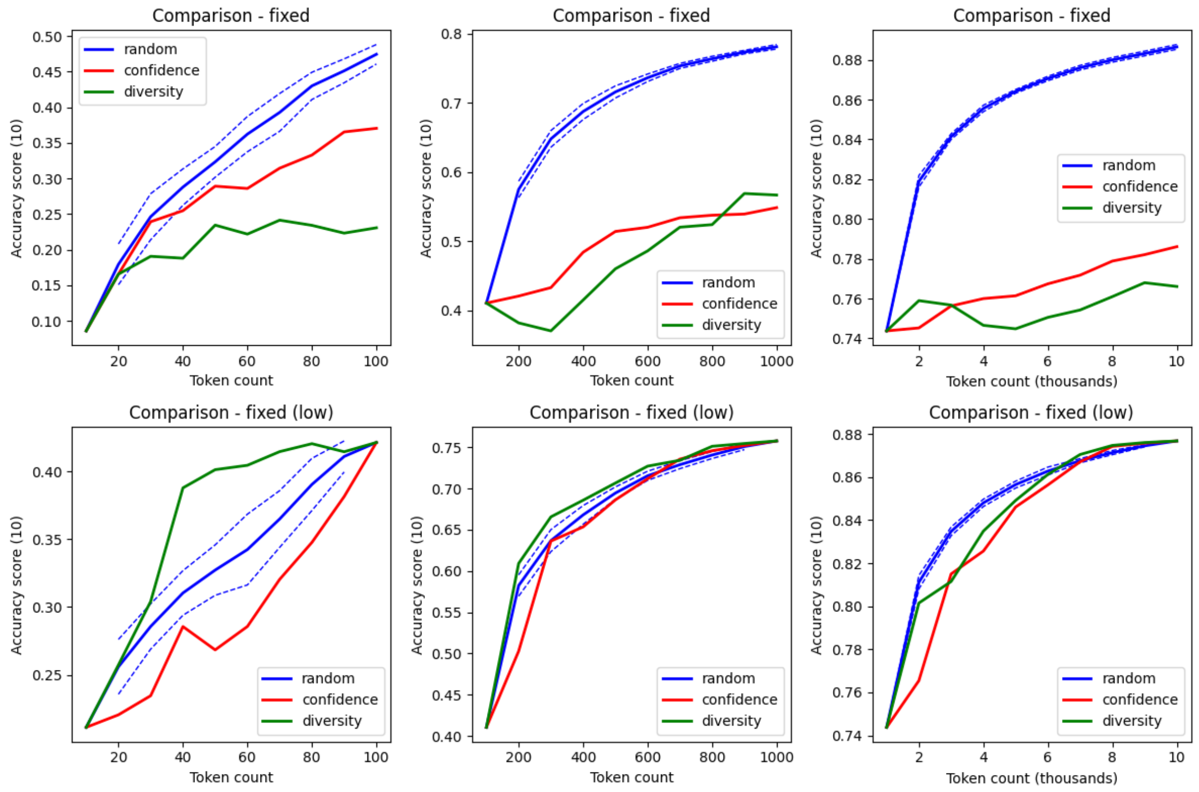

Fig.5: Comparison for 10, 100 and 100k initial tokens

At this stage we conclude that it is possible to manipulate the results simply by altering the amount of data and the way this data is structured. We can equally claim that the confidence or diversity strategy succeed and/or beat the other depending on the variant of our experiment that we select.

3.3. Accuracy

What follows is not directly related to our experiment: we want to discussed the *featured* variant with a more complex model, as well as the *oracle* strategy. Neither, however, changes our previous observations. What interests us here is the accuracy score.

Our entire *dataset* and, as a result, our *reference dataset* rely on labels that have been automatically annotated. This means that any accuracy score shown in this experiment is against an artificial reference (whose model has an accuracy measured at 0.97-98 against an actual, manually-revised reference dataset). Still, the scores we obtained are worth a mention.

Figures 4 and 5 have shown the accuracy of a single-feature model, essentially relying on token position only. For that model, the expected accuracy reaches ~0.45 for 100 tokens, ~0.78 for 1k tokens, ~0.89 for 10k tokens and ~0.94 for 100k tokens. Those scores suggest that, even against real data, a model (with 100k tokens) should not go under 0.92 accuracy. We wanted to see if adding features would affect the curves – it did not – but also, how much it would affect the accuracy.

Figure 6 is the result. It compares the random and confidence strategies on a featured model; we added an 'oracle' strategy that, again, replaced the file's confidence score with an accuracy score. On a fixed variant, the relation between confidence and accuracy is especially visible. Still, this oracle strategy – along with re-training the model on each added file and keep the best score – fails to beat a random one.

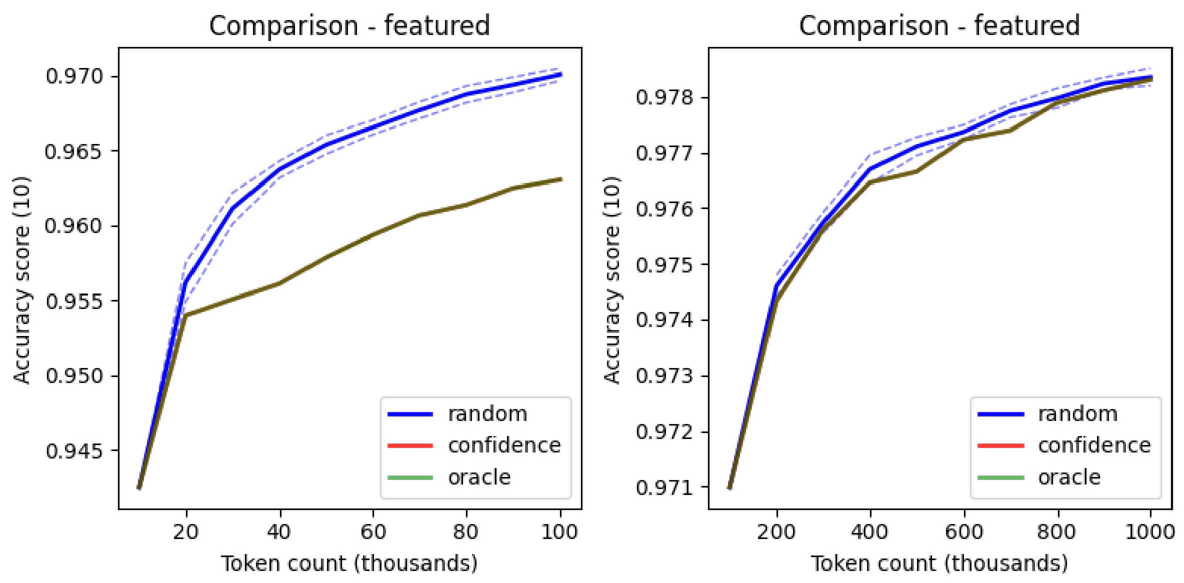

Fig.6: Learning curves with added features

Accuracy, however, reaches ~0.97 for 100k and almost ~0.98 at 1mn (one million) tokens. At that scale, we can observe the curves joining again: this is due to half of the available data being exhausted (2mn – 100k reference – 1mn subset). This suggests that a decent model on real data should not go under 0.94 and that a single-layer CRF model, with the right features, can probably reach 0.98.

Conclusion

It is possible to for an active strategy to beat a random selection. What that strategy is depends on how advanced the subset is and whether it will actually be better depends, in a large part, on how much data remains.

We can see why reference datasets tend to be in the 100k tokens range: gains past that amount become prohibitively expensive. To select that amount through an active strategy, the best candidate would be relying on the file's confidence: the gain, however, would be marginal at best. For this data and at this scale, selecting files at random might be considered optimal.

Above all, how much *available data* there is can severely affect the results, as is especially visible in figure 5 where the diversity strategy shifts from worst to best up to 1,000 tokens.

Active learning really only makes sense if data is scarce.

Finally, and beyond the scope of this experiment, we can observe how even simple, single-layer models can reach an accuracy close to multi-layered ones. This of course needs to be confirmed on real data – meaning a manual annotation – but would discourage using dictionaries or layers of CRFs. Above all it suggests that the accuracy score alone can’t be relied on to assess a model: when 0.94 is the floor and 0.98 the base expectation, the metric needs to be refined.

References

Arora Shilpa & Agarwal Sachin (2007). Active Learning for Natural Language Processing. *Language Technologies Institute School of Computer Science Carnegie Mellon University 2*, 2007.

Avanzi Mathieu, Béguelin Marie-José, Corminboeuf Gilles, †Diémoz Federica & Johnsen Laure Anne (2012-2025). *Corpus OFROM – Corpus oral de français de Suisse romande*. Université de Neuchâtel, <ofrom.unine.ch>.

Boersma Paul & Weenink David (2025). *Praat: doing phonetics by computer [Computer program]*. <praat.org>.

Chaudhary Aditi, Anastasopoulos Antonios, Sheikh Zaid, & Neubig Graham. (2021). Reducing confusion in active learning for part-of-speech tagging. T*ransactions of the Association for Computational Linguistics* 9, 1–16. DOI: 10.1162/tacl_a_00350.

Christodoulides George, Avanzi Mathieu & Goldman Jean-Philippe (2014). DisMo: A Morphosyntactic, Disfluency and Multi-Word Unit Annotator. An Evaluation on a Corpus of French Spontaneous and Read Speech. *LREC 2014*. hal-01703495.

Sutton Charles & McCallum Andrew (2010). An Introduction to Conditional Random Fields. *arXiv preprint*, 1011.4088.