ACA: the Translation Bureau's Assistant Client Advisor

Simard, Michel; Bernier-Colborne, Gabriel; Goutte, Cyril; Léger, Serge; Tessier, Marc

https://doi.org/10.4224/40003459

NATIONAL RESEARCH COUNCIL CANADA

ACA: The Translation Bureau's Assistant Client advisor

Final Report on the Domain Prediction Management Environment

Michel Simard Gabriel Bernier-Colborne
Cyril Goutte Serge Léger Marc Tessier

March 31, 2023

Executive Summary

This document reports on project *Document Workflow ("ACA")*, part of the BtB-NRC Collaboration Agreement 2021-22, titled *Artificial Intelligence for Translation Quality* (AI4TQ).

This project is about building computer support tools for the Translation Bureau's client advisors, more specifically to identify the specialty domain of texts submitted for translation. In a previous project, we used Bureau data to create classifiers that can identify to which domain a given document belongs, with an accuracy close to 80%. We delivered a system to the Bureau, called the *Assistant Client advisor* (ACA), which provides document classification as a web service, accessible both through a web-based user interface (UI) and an Application Programming Interface (API).

In this project, we have greatly expanded this system, by developing a set of functionalities that allow creating, updating, evaluating and deploying domain predictors. The API and UI of this new system will allow the Bureau to create and maintain domain predictors themselves.

In addition, we have experimented with approaches to improve prediction accuracy, most notably through neural networks. The new API allows creating and using predictors based on the *FastText* neural network technology, in addition to the algorithms previously available, SVM and ProbCat.

In a series of experiments on Confidence Estimation, we have analyzed the performance of the classifiers, and the relationship between classification accuracy and some numerical indicators produced by classifiers, with the goal of distinguishing between documents that can be handled automatically and documents that should be verified by a client advisor, with the goal of minimizing domain prediction errors and human workload.

Finally, we have added functionalities to segment large documents into smaller pieces, based on the predicted domain of individual segments of text.

Contents

Introduction

This document reports on project *Document Workflow ("ACA")*, part of the BtB-NRC Collaboration Agreement 2021-22, titled *Artificial Intelligence for Translation Quality* (AI4TQ).

This project is about building computer support tools for the Translation Bureau's client advisors (CA's), more specifically to identify the specialty domain of texts submitted for translation. It is a follow-up to the Assisted Translation Assignment (HF4) project [Goutte et al., 2020], which resulted in a prototype "Assistant Client Advisor" (ACA) application that predicts document specialty ("domain"). In that project, we have used Bureau data to create classifiers that can identify to which domain a given document belongs, with an accuracy of around 80%.

Work within this project expands on this previous work along four axes:

Continuous Training and Classifier Updates: In an operational setting, the classifiers doing domain prediction need to be updated periodically, to account for evolving domain vocabulary, and possibly changes in the domain structure. One of the goals of this project was to identify and implement optimal classifier maintenance strategies. This axis of work has lead to the development of an extended API, that allows the user of our ACA system to create, update and generally manage domain predictors.

Document Segmentation: Some larger documents contain subparts that belong to different domains. Currently, the segmentation of large documents must be performed manually. We have examined how the ACA's classifier can be used for this purpose. The new system includes functionalities that can be used to segment a large document into manageable sub-parts, that can be assigned to different resources.

Neural Network-based Domain Prediction: The previous ACA application relied on two different machine learning technologies to produce domain predictors: Support Vector Machines (SVM) and a variant of Naive Bayes (*ProbCat*). Within this project, we have explored the potential of

neural networks for this task. More specifically, we have experimented with the FastText technology [Bojanowski et al., 2017, Joulin et al., 2016, Joulin et al., 2017]. The new version of the ACA makes it possible to create and use FastText classifiers.

Confidence Estimation for Domain Prediction: We have examined the relationship between the numerical scores produced by ACA classifiers and the accuracy of their predictions. We have found that these values can be used to distinguish between cases that can be handled completely automatically and cases that must be verified by a human advisor, with the goal of minimizing both domain prediction errors and human workload.

This report is structured as follows: After a short overview of the context of this project and a summary of the previous project's outcomes, we detail each axis of work in the subsequent sections. We conclude with some potential directions for future work.

1 Context

1.1 Project Management at the Bureau

Every year, the Translation Bureau handles over 250,000 jobs submitted by its clients within federal government. Each job relates to one of the various services the Bureau offers: revision, editing, terminology, etc. But over 60% of jobs are requests for translation between Canada's two official languages. Each job is assigned to one of the Bureau's 80 "client advisor" (CA), the people responsible for overseeing and guiding the process.

Most translation jobs arrive through the Bureau's *Online Ordering System*: there, clients submit their documents and specify what is needed and when. From there, each submitted job needs to be analyzed by a CA and routed to the appropriate services. Because a large proportion of jobs are for short documents – 47% of French documents and 54% of English documents have less than 250 words – this management task is quite costly for the Bureau.

Every translator have their areas of specialty: finance, health sciences, legal, etc. The Bureau recognizes 20 areas of specialty and 64 sub-specialties. This information is associated with the translator's profile. The CA's decision to assign a job to a given translator is partly based on this information. For this reason, they need to know the areas of specialty and sub-specialty and be able to recognize them efficiently.

An important proportion of documents submitted for translation are known as "General Administrative Texts", in French "Textes administratifs et généraux" (*TAG*) : 55% of English and 56% of French translations. These can be handled by any translator, in-house or freelance. All the rest need to be handled by an appropriate specialist.

While most documents can be handled by a single translator, there are occasional documents that touch on more than one area of specialty. For example, a contract for computer hardware may contain a "technical" section and a "legal" section. As a result, CA's cannot simply scan the first few pages of a document, they need to analyze the complete document. When there are multiple domains, the CA needs to segment the document, so that each segment can be assigned to a different resource.

1.2 Outcome of the *Assisted Translation Assignment* Project (2019-2021)

In a previous project, we have exploited some of the Translation Bureau's databases to create custom *text domain predictors*: these systems take as input the text of a document, and possibly some metadata, such as the name of the client organization, and outputs the name of a specialty domain. We experimented with two different machine learning classification approaches: Support Vector Machines (SVM) [Hearst et al., 1998] and a variant of Naive Bayes, which we call ProbCat [Goutte et al., 2014]. We tested monolingual (English-only and French-only) and bilingual (English-French) classifiers. We also compared straightforward, one-step classification with a two-step approach, that first tries to distinguish between general (TAG) and specialized documents, then establish the specialty domain for the latter.

In practice, bilingual classifiers performed better, and two-step classification produced more useful results. While SVM proved more accurate generally, we found that ProbCat produced a more useful ranking of domains in the second step. Our best systems correctly distinguished general from specialized documents 78% of the time, and assigned over 85% of specialized documents to the correct domain.

We delivered the best predictors – SVM and ProbCat, one-step and two-step – to the Bureau by packaging them into a web service, available through an API as well as a simple web user interface. This work and the deliverables are described in detail in the project's final report [Goutte et al., 2020].

2 Classifier Training and Management

In an operational setting, classifiers doing domain prediction need to be updated periodically, to account for evolving domain vocabulary, and possibly changes in the domain structure. To make it possible for Bureau personnel to perform this maintenance work themselves, we developed new functionalities and extended the API of the web service and the user interface accordingly. We describe these changes in this section.

2.1 Application Programming Interface (API)

Having the ability to train classifiers from scratch and in a continuous fashion requires that users be able to provide the system with training data. As the system is designed as a web service, this has important implications. The extended API has functions for data sets, and functions for classifiers.

The basic unit for managing data is the **dataset**. Each *dataset* object has a *static* part and a *dynamic* part: Static data is uploaded when the dataset is created, and is used to train a predictor "from scratch". Dynamic data is used for continuous updates. By default, when adding new data to an existing dataset, this gets added to the dynamic part. At any point, dynamic data can be "consolidated", i.e. merged with the static, or deleted. This is illustrated in Figure

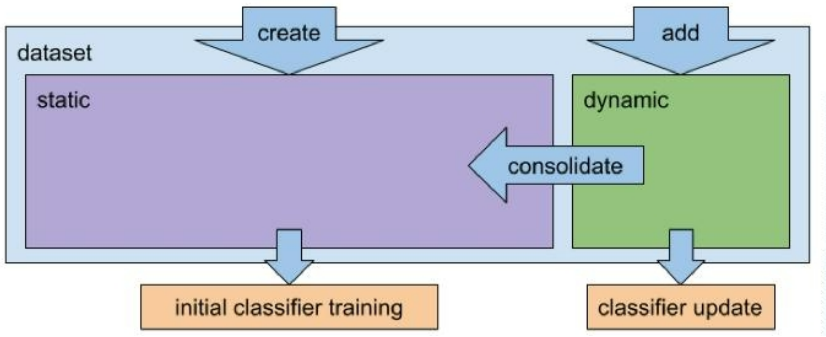

Figure 2.1: The *dataset* object.

2.1

Datasets can also be used to test (evaluate) classifiers. In that case, both static and dynamic parts of the dataset are used: the data is submitted to the classifier, and the predicted domain is compared to the real domain as recorded in the dataset to compute performance metrics.

The API contains functions to:

- list existing datasets
- create a new dataset, providing data entries in JSON or CSV format. By default, data gets added to the "dynamic" part; but it is possible to specify the "static" part with a parameter.
- add dynamic data to a dataset (JSON or CSV)
- consolidate data: this means transferring all "dynamic" entries to the "static" part.
- create a copy of a dataset (for example, to create a backup)
- download the content of a dataset as a CSV file.
- delete all dynamic data in a dataset (while retaining static data)
- delete the complete dataset.

Then, there are *classifier* objects, which implement domain predictors. A classifier object is associated with a training *dataset* right from the start.

Classifiers support three classification algorithms: SVM (Support Vector Machines), ProbCat (Naive Bayes) and FastText. Each algorithm has advantages and disadvantages. Section 4 discusses these in more detail.

Domain prediction can follow one of two different strategies:

- one-step classification: this is a straightforward approach, where the classifier directly predicts the most likely specialization domain for a given document.
- two-step classification: in this approach, a first predictor determines whether or not the document belongs to the general administrative domain (TAG). If not, then a second predictor determines the most likely (non-TAG) specialized domain.

When using a two-step strategy, we actually need two classifier objects, which we refer to as *Phase1* (TAG or non-TAG) and *Phase2* (which specialized domain). One-step classifiers consist of a single classifier which, by analogy, we call *Phase0*.

Training a classifier creates a *model* object within the classifier. In practice, multiple *models* can co-exist within the same classifier, each corresponding to a distinct training run.

The *classifier* API contains functions to:

- list all existing classifiers;
- create a new classifier, by specifying a dataset, a type (Phase0, Phase1 or Phase2) and an algorithm.
- train a classifier; this uses the static data from the associated dataset and produces a classifier model.
- update a classifier model: this uses the dynamic data from the associated dataset
- predict the domain of a document using a model
- evaluate a model: this applies the model to a test dataset and reports

various metrics.

- deploy a model to production: this identifies a given classifier model (or a pair of models, when doing two-step classification) as the default for predictions.
- delete individual models or an entire classifier

The current API doesn't allow to "undo" an operation. For example, once a dataset has been consolidated, it's not possible to remove the data that was just moved from the dynamic to the static part; once a model has been trained or updated, it's not possible to revert to the previous state. However, there are "copy" functions for all objects (datasets, classifiers, models). This makes it possible to create back-ups of objects at critical points.

With this model, the life of a classifier might look something like this. Assuming we have some initial data, that we split into a training set and a test set, which we store in two CSV files `my-training-data.csv` and `my-test-data.csv`.

- Create a dataset D, uploading `my-training-data.csv` to D's static part
- Create a classifier C (e.g. a *Phase0* SVM), associated with dataset D
- Train C: this uses D's static data.
- Create another dataset T for testing, uploading data from file `my-test-data.csv`
- Test C: run it on T to compute its accuracy – let's call this A_0
- Repeat:
  - Add dynamic data D_k to D
  - Update (retrain) classifier C, using D's dynamic data
  - Re-evaluate C on $T \rightarrow A_k$
  - Check that the accurracy has not degraded: $A_k \geq A_{k-1}$
  - Consolidate D

Periodically, we might want to retrain from scratch using all available data. For that, we just need to "consolidate" dataset D, then train C. Also, if we

think we might want to backtrack if performance ever degrades, we can create a copy of *C* before performing the update.

All the technical details about the API are available online through the ACA web service, at `https://<ACA-server-address>/docs`.

2.2 Web User Interface

The new version of the ACA comes with an expanded Web-based User Interface (UI). While this is primarily intended as a demo, it implements the more common operations enabled by the API. Details about this can be found in the document entitled *ACA Web User Interface*.

3 Document Segmentation

While most documents can be handled by a single translator, there are occasional documents that touch on more than one area of specialty. For example, a contract for computer hardware may contain a "technical" section and a "legal" section. As a result, CA's cannot simply scan the first few pages of a document, they need to analyze the complete document. When there are multiple domains, the CA needs to segment the document, so that each segment can be assigned to a different resource.

We have examined how the ACA's classifier can be used for this purpose. The approach we have considered is one in which a document is first segmented into a series of "chunks", each chunk is submitted for domain prediction, and the final segmentation is obtained by aggregating adjacent chunks that belong to the same domain.

Unfortunately, the data that we had access to for this project (the DC-23 database and the 2014-2018 Megacorpus, see [Goutte et al., 2020]) did not contain the kind of detailed information necessary to evaluate the quality of the segmentations produced with this approach. All documents in our data set are assigned to a single specialization domain. Furthermore, the Megacorpus

does not contain any structural information about the documents: each document is viewed as a simple list of "translation units", usually sentences.

Nevertheless, the ACA `predict` function now allows to specify an initial segmentation of the document into chunks. When this is provided, in addition to the normal document-level domain prediction, the system produces a list of chunk-level predictions for each input chunk.

The API also contains a function `segment` which, given the text of a document in plain text format, will produce a segmentation of the text into chunks, which can be provided to the `predict` function as the initial chunking. The `segment` function also accepts an argument to specify the minimum number of words per chunk. While this functionality may be useful to produce a segmentation when no other information is available, it is much preferable to base the segmentation on the structure of the document, for example by considering whole sections or chapters.

It is also worth noting that prediction accuracy can be expected to decrease as chunks get smaller. Although we do not have precise measurements for this, we suspect that predictions computed on chunks of less 250 words are not likely to be very reliable.

4 Neural Networks for Domain Prediction

The previous ACA application relied on two different machine learning technologies to produce domain predictors: Support Vector Machines (SVM) and a variant of Naive Bayes (*ProbCat*). Both these methods can be considered "old" by current standards of machine learning research. Within this project, it made sense to explore the potential of more "modern" approaches, such as those based on neural networks.

More specifically, we have experimented with the FastText technology [Bojanowski et al., 2017, Joulin et al., 2016, Joulin et al., 2017]. Our objectives were to find the best training configuration for FastText, evaluate its ability to assign the correct domain labels to source documents that were translated by

the Translation Bureau (BTB), and compare those results to those previously obtained with SVM and ProbCat classifiers. We also wished to compare single-step and two-step classification (i.e. predict TAG first, then other classes).

In our experiments, the best results were obtained when combining English and French data to create a single one-step classifier (as opposed to two-step, *TAG-vs.-all* followed by *all-but-TAG* classification) and using the client name as a distinct feature (in addition to the document text).

Overall, FastText performed slightly better than ProbCat and SVM on French texts, but not as well as ProbCat on English. SVM came third in both cases. Scores were very similar in both languages, but highly variable across classes. The length of the text did not correlate with accuracy. Most classification errors were between TAG and other classes. Other errors mostly involved similar domains (e.g. SCN, BIO, and ENV). In the end, we found that FastText did not provide substantial gains in accuracy, and was more expensive to train and more complicated to optimize than ProbCat and SVM.

More details about our experiments and findings can be found in technical report [Bernier-Colborne and Simard, 2022]. The new version of the ACA makes it possible to create and use FastText classifiers.

5 Classification Confidence

Classification algorithms typically associate numerical scores with their predictions. These scores can inform the user on the reliability of the systems' decisions. As part of this project, we have analyzed the scores produced by classifiers and the link between these scores and the accuracy of the predicted domains. In this section, we discuss how these scores can be used to distinguish between cases that can be handled completely automatically and cases that a human advisor should verify.

The ACA implements different classification strategies and algorithms (see Section 2): supported algorithms include SVM, Naive Bayes ("ProbCat") and FastText; strategies include one-step and two-step classification. Whatever the

algorithm and strategy, when the system is used to predict the domain of a new document, it always produces a score alongside the predicted domain. Table 5.1 shows some examples of such scores, in this case, produced by a 1-step FastText classifier.

Doc. ID	lang.	client	text	domain	score
9038263	en	333	*Hello everyone, It is with pleasure that...*	TAG	0.996391
9038395	en	001	*AAFC's email is changing July 2015! ...*	TAG	0.999960
9061992	en	040	*The SST GD issued a direction...*	EMP	0.972531
9071172	en	349	*Listing of Work Place Hazards and...*	TECH	0.243826
9077132	en	349	*INSTRUCTIONS FOR WRITTEN EXAMINATION...*	EMP	0.663619

Table 5.1: Examples of document domain predictions and scores.

The exact meaning of the score depends on the classification algorithm that produced it. In some cases, they represent a probability estimate, other times they are confidence levels. But in all cases, in the ACA implementation, they have similar behaviours: they are always real-valued numbers, with values that vary between 0 and 1, and higher values denoting more reliable predictions ("Bigger is better").

We consider a scenario where documents are routed differently, based on the values of the ACA prediction scores: documents whose domain prediction score falls below a given threshold are considered "unsure", and redirected to a human client advisor for manual validation; all others are considered reliable enough to go through without human supervision. This approach is not entirely foolproof: in practice, we observe that many low-scoring predictions nevertheless turn out to be correct, and conversely, some high-scoring predictions are wrong. But it is possible to set thresholds on scores, in such a way as

to target a specific level of error.

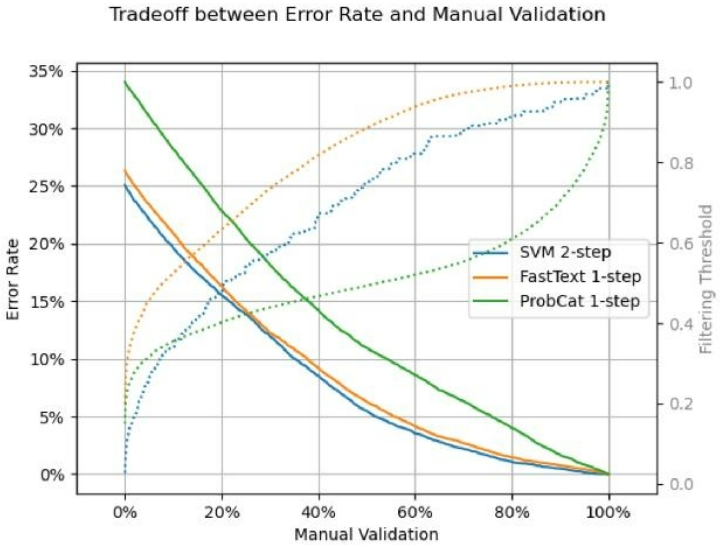

Figure 5.1: Domain classification error rate, as a function of the percentage of documents routed to manual validation, using thresholding on prediction scores (corresponding score thresholds are shown with dotted lines). Error rates are computed under the assumption that manual classifications are always correct.

Figure 5.1 shows how the expected classification error rate varies in such a setup, as a function of the percentage of document classifications that are validated manually, on a sample of 10,000 documents. We show this for three typical setups: 1-step ProbCat (solid green line), 1-step FastText (solid orange) and 2-step SVM (solid blue). Here, we are making the assumption that humans do not make errors when assigning documents to domains. Therefore, the expected error rate when 100% of documents are sent to manual validation is zero, as can be seen on the right side of the chart. At the opposite end (left side of the chart), when no document is validated manually, the expected error rate is that of the used ACA classifier itself. For example, for the SVM 2-

step classifier displayed here, this would be approximately 25%. By assigning a fraction of documents to manual validation, it is possible to lower this error rate to a specific value. For example, again using the SVM 2-step classifier, manually validating only the 40% lowest-scoring documents allows lowering the error rate to approximately 8%.

Dotted lines in Figure 5.1 indicate the scoring threshold corresponding to the percentage of documents that go to manual validation (note that the corresponding scale is on the right-hand side of the chart). For example, to assign 40% of documents to manual validation, the score threshold for the 2-step SVM should be set to 0.66. For the 1-step FastText, this threshold would be 0.815; and for the 1-step ProbCat, it would be 0.43.

This last example highlights how the scores produced by different classification algorithms can have very different behaviours. Figure 5.2 shows the distribution of scores produced by SVM, ProbCat and FastText classifiers on our sample of 10,000 documents.

In practice, filtering thresholds must be determined empirically for each classifier, using a method similar to what was done to produce the chart in Figure 5.1: Working with a representative sample of manually labelled documents, distinct from those used to train the classifiers, we compute predictions, identify errors, then use that information to determine the threshold on the score. In our experiments, we used 10,000 documents – we recommend using at least 2000. In a future version of the ACA, this process could possibly be automated.

6 Conclusions and Future Work

The main goal of the ACA project was to develop technologies to support the routing of documents at the Translation Bureau. While our work touched on various topics related to this goal, the main outcome of the project is the new expanded API for the ACA system, which we believe will allow the Bureau to create and maintain its own domain predictors. The API was designed to support the most common operations while relieving the users of most of the te-

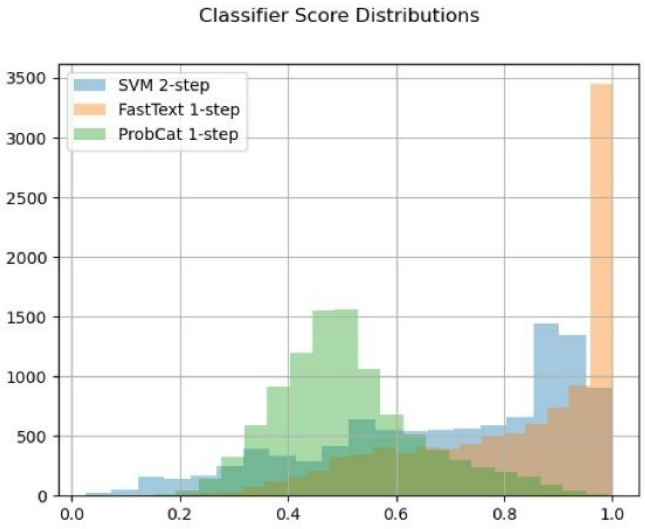

Figure 5.2: Distribution of prediction score values for different classification algorithms in the ACA.

dious details normally involved in deploying this kind of technology. It is quite possible that some functionalities are missing and that some design choices will need to be revisited. We are looking forward to feedback from users to help us tailor the system to their needs.

For example, we recognize that managing data sets is an important part of the work required when developing AI applications. With this in mind, data management functions could be added to split data into training, tuning and testing sets automatically. We could also examine the possibility of providing data analysis tools to help detect noisy entries, outliers, inconsistencies in the annotations, etc. Also, our work on confidence estimation suggests that some aspects of performance analysis could be automated, so as to facilitate such tasks as identifying problematic documents and determining thresholds for manual validation, etc. Our work on text segmentation highlighted the need

for allowing users to specify potential segmentation points in document data. This could be achieved, for example, through rudimentary structural markup in the input of the predictors.

On the research front, text classification is a constantly evolving field and new techniques are proposed regularly. We would like to evaluate the potential of some newer approaches, such as so-called Large language models (LLM), that have proved fruitful in related NLP tasks. We would also like to examine whether terminological resources, such as Termium Plus, which has its own domain structure, etc. could potentially be used to produce additional features to predict the domains of the Bureau.

However, at this point, we feel that improvements in classification accuracy may depend less on algorithm selection and tuning, and more on extrinsic factors, such as the reliability of the labels, the separability of classes, class imbalances, data preprocessing (segmentation, tokenization, case normalization, etc.). Data collection, revision and analysis may be more fruitful avenues than further experiments in model selection and tuning. Another interesting question is whether the automatic clustering of documents could give insights into ways of revising the set of domains in a data-centric fashion.

References

[Bernier-Colborne and Simard, 2022] Bernier-Colborne, G. and Simard, M. (2022). Domain Prediction Using FastText. Technical report, National Research Council Canada.

[Bojanowski et al., 2017] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[Goutte et al., 2014] Goutte, C., Léger, S., and Carpuat, M. (2014). The nrc system for discriminating similar languages. In *Proceedings of the first workshop on applying NLP tools to similar languages, varieties and dialects*, pages 139–145.

[Goutte et al., 2020] Goutte, C., Simard, M., and Léger, S. (2020). Gestion assistée des commandes au Bureau de la traduction. Technical report, National Research Council Canada.

[Hearst et al., 1998] Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

[Joulin et al., 2016] Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016). Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

[Joulin et al., 2017] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.