# Comparing Feature-Engineering and Feature-Learning Approaches for Multilingual Translationese Classification

**Daria Pylypenko**[*1], **Kwabena Amponsah-Kaakyire**[*1,2], **Koel Dutta Chowdhury**[*1],
**Josef van Genabith**[1,2], and **Cristina España-Bonet**[2]
[1]Saarland University, [2]German Research Center for Artificial Intelligence (DFKI)
Saarland Informatics Campus, Saarbrücken, Germany
{daria.pylypenko,koel.duttachowdhury}@uni-saarland.de
kwabena.amponsah-kaakyire@dfki.de
{cristinae, Josef.Van_Genabith}@dfki.de

## Abstract

Traditional hand-crafted linguistically-informed features have often been used for distinguishing between translated and original non-translated texts. By contrast, to date, neural architectures without manual feature engineering have been less explored for this task. In this work, we (*i*) compare the traditional feature-engineering-based approach to the feature-learning-based one and (*ii*) analyse the neural architectures in order to investigate how well the hand-crafted features explain the variance in the neural models' predictions. We use pre-trained neural word embeddings, as well as several end-to-end neural architectures in both monolingual and multilingual settings and compare them to feature-engineering-based SVM classifiers. We show that (*i*) neural architectures outperform other approaches by more than 20 accuracy points, with the BERT-based model performing the best in both the monolingual and multilingual settings; (*ii*) while many individual hand-crafted translationese features correlate with neural model predictions, feature importance analysis shows that the most important features for neural and classical architectures differ; and (*iii*) our multilingual experiments provide empirical evidence for translationese universals across languages.

## 1 Introduction

Texts originally written in a language exhibit properties that distinguish them from texts that are the result of a translation into the same language. These properties are referred to as *translationese* (Gellerstam, 1986). Earlier studies have shown that using various hand-crafted features for supervised learning can be effective for translationese classification (Baroni and Bernardini, 2005; Volansky et al., 2015; Rubino et al., 2016). However, this approach has a number of limitations. Firstly, manually designed features may be partial and non-exhaustive in a sense that they are based on our linguistic intuitions, and thus may not be guaranteed to capture all discriminative characteristics of the input data seen during training. Other limitations are related to the difficulties in obtaining linguistic annotation tools (e.g., parsers, taggers, etc.) for some languages, reliance on $n$-gram counts, limited contexts, corpus specific characteristics, among others. In this work, we compare a standard approach based on hand-crafted features with automatic feature learning based on data, task and learner without prior linguistic assumptions.

Moreover, most previous approaches have focused on classifying translationese in the monolingual setting, i.e. translations come from one or multiple source languages, but the language on which to perform the classification is always the same. To the best of our knowledge, the multilingual setting with multiple source and target languages has not been explored yet. If translationese features are language-independent or shared among languages, multilingual translationese classification experiments would show the effect. We perform binary translationese classification not only in mono-, but also in multilingual settings to empirically verify the existence of translationese universals throughout different source and target languages.

In our work we investigate:

(*i*) How automatic neural feature learning approaches to translationese classification compare to classical feature-engineering-based approaches on the same data. To do this, we use pre-trained embeddings as well as several end-to-end neural architectures.

(*ii*) Whether it is possible to effectively detect translationese in multilingual multi-source data, and how it compares to detecting translation in monolingual and single-source data in different languages.

[*]Equal contribution.

*Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8596–8611
November 7–11, 2021. ©2021 Association for Computational Linguistics

(*iii*) Whether $a$) translationese features learned in one setting can be useful in a different setting and $b$) the overhead of training separate monolingual models can be reduced by either multi-source monolingual models for a given target language or even better, a multilingual model. For this we perform cross-data evaluation.

(*iv*) Whether variation observed in predictions of neural models can be explained by linguistically inspired hand-crafted features. We perform linear regression experiments to study the correlation between hand-crafted features and predictions of representation learning models as a starting point for investigating neural models which do not lend themselves easily to full explainability.

We show that:

- representation-learning approaches outperform hand-crafted feature-selection methods for translationese classification, with BERT giving the highest accuracy,
- it is possible to classify translationese in the multilingual data, but models trained on monolingual single-source data generally yield better performance than models trained on multi-source and multilingual data,
- in contrast to hand-crafted feature-based models, neural models perform relatively well on different datasets (cross-data evaluation), and single-source can, to a reasonable extent, be substituted by multi-source mono- and multilingual models,
- many traditional hand-crafted translationese features exhibit significant correlation with the predictions of the neural models. However, a feature importance analysis shows that the most important features for neural networks and for classical architectures differ.

The paper is organized as follows. Section 2 describes related work. Section 3 introduces the architectures used in our study. Section 4 discusses the data and presents the main classification results. We perform cross-data evaluations in Section 5 and analyze feature importance and correlation in Section 6. Finally, we summarize and draw conclusions in Section 7.

## 2 Related Work

Recent work on translationese, both on human- and machine-translated texts, explores topics ranging from translationese characterization (Volansky et al., 2015; Bogoychev and Sennrich, 2019; Bizzoni et al., 2020) to unsupervised classification (Rabinovich and Wintner, 2015), to exploring insights into the structure of language typologies with respect to different translationese properties (Rabinovich et al., 2017; Bjerva et al., 2019; Dutta Chowdhury et al., 2020, 2021), to the effects on downstream tasks such as machine translation (Stymne, 2017; Toral et al., 2018; Zhang and Toral, 2019; Freitag et al., 2019; Edunov et al., 2020; Riley et al., 2020; Graham et al., 2020), to translationese data collection (Rabinovich et al., 2015; Nisioi et al., 2016; Amponsah-Kaakyire et al., 2021).

Traditional translationese classification approaches rely on manually designed features, such as $n$-gram frequencies on tokens, part-of-speech (POS) tags or lemmas (Baroni and Bernardini, 2005; van Halteren, 2008; Kurokawa et al., 2009), function word frequencies (Koppel and Ordan, 2011; Tolochinsky et al., 2018), character-level features (Popescu, 2011; Avner et al., 2016), surface and lexical features (Ilisei et al., 2010; Volansky et al., 2015), syntactic features (Ilisei et al., 2010; Rubino et al., 2016), morpheme-based features (Avner et al., 2016; Volansky et al., 2015), information-density-based features (Rubino et al., 2016), etc.

By contrast, to date neural approaches to translationese (Bjerva et al., 2019; Dutta Chowdhury et al., 2020) have received less attention. While Bjerva et al. (2019) have used learned language representations to show that the distance in the representational space reflects language phylogeny, Dutta Chowdhury et al. (2020, 2021) use divergence from isomorphism between embedding spaces to reconstruct phylogenetic trees from translationese data. Sominsky and Wintner (2019) train a BiLSTM for translation direction identification and report accuracy up to 81.0% on Europarl data.

## 3 Architectures

### 3.1 Feature-Selection-Based Classification (`Handcr.+SVM`)

We employ the *INFODENS* toolkit (Taie et al., 2018) to extract hand-crafted features to train and evaluate a classifier. We use a support vector machine classifier (SVM) with linear kernel, and fit the hyperparameter $C$ on the validation set. For the choice of features, we replicate the setup from (Amponsah-Kaakyire et al., 2021), using a 108-dimensional feature vector, inspired by the feature set described in (Rubino et al., 2016). In particular, we use:

1. **surface features**: average word length, syllable ratio, paragraph length. These surface features can be connected to the simplification hypothesis (Ilisei et al., 2010; Volansky et al., 2015), as it is assumed that translations contain simpler shorter words than original texts.
2. **lexical features**: lexical density, type-token ratio. These lexical features can also be linked to the simplification hypothesis, due to the assumption that original texts have richer vocabulary than translated ones and contain a higher proportion of content words (Laviosa, 1998; Baker et al., 1993).
3. **unigram bag-of-PoS**: These features correspond to the source interference (shining-through) hypothesis (Volansky et al., 2015), as POS $n$-grams reflect grammatical structure, which might be altered in translations due to the influence of the source language grammar.
4. **language modelling features**: log probabilities and perplexities with and without considering the end-of-sentence token, according to forward and backward $n$-gram language models ($n \in [1; 5]$) built on tokens and POS tags. It is hypothesized that the perplexity of translated texts may be increased because of simplification, explicitation and interference (Rubino et al., 2016).
5. **$n$-gram frequency distribution features**: percentages of $n$-grams in the paragraph occurring in each quartile ($n \in [1; 5]$). This feature could be linked to the normalization hypothesis, according to which translated texts are expected to contain more collocations, i.e. high-frequency $n$-grams (Toury, 1980; Kenny, 2001).

In our experiments, language models and $n$-gram frequency distributions are built on the training set. The $n$-gram language models are estimated with SRILM (Stolcke, 2002) and SpaCy[1] is used for POS-tagging. Features are scaled by their maximum absolute values. The full list of 108 features is given in the Appendix A.1.

## 3.2 Embedding-based Classification

### 3.2.1 Average pre-trained embeddings + SVM (`Wiki+SVM`)

We compute an average of all token vectors in the paragraph, and use this mean vector as a feature vector to train a SVM classifier with linear kernel. We work with the publicly available language specific 300-dimensional pre-trained Wiki word vector models trained on Wikipedia using *fastText*[2] (Joulin et al., 2016).

### 3.2.2 Gaussian distributions for similarity-based classification (`Wiki+Gauss.+SVM`)

We follow Das et al. (2015); Nikolentzos et al. (2017) and Gourru et al. (2020) and represent a text as a multivariate Gaussian distribution based on the distributed representations of its words. We perform similarity-based classification with SVMs where the kernel represents similarities between pairs of texts. We work with the same pre-trained Wikipedia embeddings as in `Wiki+SVM` for the words in the model and initialize the ones not contained in the model to random vectors.

Specifically, the method assumes that each word `w` is a sample drawn from a Gaussian distribution with mean vector $\mu$ and covariance matrix $\sigma^2$:

$$\mathtt{w} \sim \mathcal{N}(\mu, \sigma^2) \tag{1}$$

A text is then characterized by the average of its words and their covariance. The similarity between texts is represented by the convex combination of the similarities of their mean vectors $\mu_i$ and $\mu_j$ and their covariances matrices $\sigma_i^2$ and $\sigma_j^2$:

$$\textit{similarity} = \alpha(sim(\mu_i, \mu_j)) + (1 - \alpha)(sim(\sigma_i^2, \sigma_j^2)) \tag{2}$$

where $\alpha \in$ [0,1] and the similarities between the mean vectors and co-variances matrices are computed using cosine similarity and element-wise product, respectively. Finally, a SVM classifier is employed using the kernel matrices of Equation 2 to perform the classification.

[1] https://spacy.io/

[2] https://fasttext.cc/docs/en/pretrained-vectors.html

## 3.3 Neural Classification

### 3.3.1 fastText classifier (`FT`)

*fastText* (Joulin et al., 2016) is an efficient neural network model with a single hidden layer. The *fastText* model represents texts as a bag of words and bag of $n$-gram tokens. Embeddings are averaged to form the final feature vector. A linear transformation is applied before a hierarchical softmax function to calculate the class probabilities. Word vectors are trained from scratch on our data.

### 3.3.2 Pre-trained embeddings + FT (`Wiki+FT`)

In this model we work with the pre-trained word vectors from Wikipedia to initialize the fastText classifier. The data setting makes this directly comparable to `Wiki+SVM`, a non-neural classifier.

### 3.3.3 Long short-term memory network (`LSTM`)

We use a single-layer uni-directional LSTM (Hochreiter and Schmidhuber, 1997) with embedding and hidden layer with 128 dimensions. The embedding layer uses wordpiece subunits and is randomly-initialised. We pool (average) all hidden states, and pass the output to a binary linear classifier. We use a batch size of 32, learning rate of $1{\cdot}10^{-2}$, and Adam optimiser with *Pytorch* defaults.

### 3.3.4 Simplified transformer (`Simpl.Trf.`)

We use a single-layer encoder–decoder transformer with the same hyperparameters and wordpiece embedding layer as the LSTM. The architecture has no positional encodings. Instead, we introduce a simple cumulative sum-based contextualisation. The attention computation has been simplified to element-wise operations and there are no feedforward connections. A detailed description is provided in Appendix A.2.

### 3.3.5 Bidirectional Encoder Representations from Transformers (`BERT`)

We use the BERT-base multilingual uncased model (12 layers, 768 hidden dimensions, 12 attention heads) (Devlin et al., 2019). Fine-tuning is done with the *simpletransformers*[3] library. For this, the representation of the [CLS] token goes through a pooler, where it is linearly projected, and a $tanh$ activation is applied. Afterwards it undergoes dropout with probability 0.1 and is fed into a binary linear classifier. We use a batch size of 32, learning rate of $4 \cdot 10^{-5}$, and the Adam optimiser with epsilon $1 \cdot 10^{-8}$. Models were fine-tuned on 4 GPUs.

| Corpus | Training | Dev. | Test |
|---|---|---|---|
| TRG–SRC | 30k | 6k | 6k |
| TRG–ALL | 30k | 6k | 6k |
| ALL–ALL[3] | 89k | 19k | 19k |
| ALL–ALL[8] | 67k | 14k | 14k |

Table 1: Number of paragraphs in each of the datasets. Average paragraph length is around 80 tokens.

We design and compare our "lean" single-layer LSTM and simplified transformer models with BERT in order to investigate whether the amount of data and the complexity of the task necessitate complex and large networks.[4]

# 4 Translationese Classification

## 4.1 Data

We use monolingual and multilingual translationese corpora from Amponsah-Kaakyire et al. (2021) which contain annotated paragraphs (avg. 80 tokens) of the proceedings of the European parliament, the Multilingual Parallel Direct Europarl[5] (MPDE). Annotations indicate the source (SRC) and target languages (TRG), the "original" or "translationese" label, and whether the translations are direct or undefined (possibly translated through a pivot language). As texts translated through a pivot language may have different characteristics from directly translated texts, here we only use the direct translations. For the initial experiment we focus on 3 languages: German (DE), English (EN) and Spanish (ES). We adopt the following format for data description: we refer to translationese corpora (i.e. corpora where half of the data is originals, half translationese) with the "TRG–SRC" notation (with a dash): TRG is the language of the corpus, SRC is the source language, from which the translation into the TRG language was done in order to produce the translationese half. The "TRG←SRC" notation (with an arrow) denotes the result of translating a text from SRC into TRG language. We use it to refer only to the

[3] `github.com/ThilinaRajapakse/simpletransformers`

[4] The two lean architectures drastically decrease the number of core parameters. The number of parameters is 85 M for BERT, 132 k for the LSTM and 768 for the simplified transformer when the embedding layer and the classifier, which are common to the 3 architectures, are not considered.

[5] `github.com/UDS-SFB-B6-Datasets/Multilingual-Parallel-Direct-Europarl`

|  | Handcr. +SVM | Wiki +SVM | Wiki +Gauss. +SVM | fastText (FT) | Wiki +FT | LSTM | Simpl. Trf. | BERT |
|---|---|---|---|---|---|---|---|---|
| DE–EN | 71.5±0.0 | 77.7±0.1 | 67.6±0.1 | 88.4±0.0 | 89.2±0.0 | 89.5±0.4 | 89.7±0.2 | **92.4±0.2** |
| DE–ES | 76.2±0.0 | 79.4±0.3 | 68.2±0.2 | 90.9±0.0 | 91.9±0.0 | 91.9±0.2 | 91.6±0.2 | **94.4±0.1** |
| EN–DE | 67.6±0.7 | 72.5±0.2 | 64.5±0.2 | 85.1±0.0 | 85.9±0.1 | 86.8±0.5 | 85.8±0.2 | **90.7±0.1** |
| EN–ES | 70.1±0.2 | 77.5±0.4 | 67.1±0.4 | 87.6±0.0 | 88.7±0.0 | 89.1±0.3 | 89.3±0.4 | **91.9±0.4** |
| ES–DE | 71.0±0.0 | 75.7±0.4 | 70.1±0.4 | 88.4±0.0 | 89.1±0.0 | 90.2±0.2 | 90.4±0.3 | **92.3±0.2** |
| ES–EN | 66.7±0.0 | 70.1±0.3 | 67.0±0.7 | 87.0±0.1 | 87.9±0.0 | 88.8±0.4 | 88.4±0.2 | **91.4±0.3** |
| DE–ALL | 72.6±0.0 | 64.3±0.0 | 65.1±0.1 | 87.4±0.0 | 88.3±0.0 | 88.5±0.2 | 88.6±0.4 | **90.9±0.3** |
| EN–ALL | 65.3±0.0 | 64.6±0.0 | 62.5±0.1 | 82.7±0.0 | 84.4±0.0 | 84.2±0.4 | 83.8±0.3 | **87.9±0.4** |
| ES–ALL | 67.4±0.0 | 67.3±0.0 | 66.5±0.2 | 84.9±0.0 | 85.9±0.0 | 87.0±0.3 | 86.9±0.3 | **89.9±0.1** |
| ALL–ALL[3] | 58.9±0.0 | – | – | 85.0±0.0 | – | 84.4±0.3 | 84.5±0.2 | **89.6±0.2** |
| ALL–ALL[8] | 65.4±0.1 | – | – | 70.4±0.1 | – | 77.2±0.3 | 77.9±0.1 | **84.6±0.2** |

Table 2: Translationese classification average accuracy on the mono- and multilingual test sets (average and standard deviation over 5 runs).

translationese half of the corpus.

For our experiments we extract four datasets from MPDE with summary statistics in Table 1.

1. Monolingual single-source data: DE–EN, DE–ES, EN–DE, EN–ES, ES–DE, ES–EN. For each corpus, there is an equal number of translated and original paragraphs.

2. Monolingual multi-source data: DE–ALL, EN–ALL, ES–ALL. For DE–ALL, e.g., half of the data is DE original texts, and the other half contains equal proportions of DE←ES and DE←EN.

3. Multilingual multi-source data: ALL–ALL[3]. There is an equal number of originals: DE, EN and ES, which together make up 50% of the examples. The other 50% which are translated are equal proportions of DE←EN, DE←ES, EN←DE, EN←ES, ES←DE and ES←EN.

EN, DE and ES are relatively close typologically. We conduct additional experiments in order to investigate how well the classification can be performed when more and more distant languages are involved:

4. Multilingual multi-source data large: ALL–ALL[8], balanced in the same way as ALL–ALL[3], but with the addition of Greek (EL), French (FR), Italian (IT), Dutch (NL) and Portuguese (PT).

For all settings we perform binary classification: original vs. translated.

### 4.2 Results

Paragraph-level translationese classification results with mean and standard deviations over 5 runs are reported in Table 2. Overall, the BERT model outperforms other architectures in all settings, followed closely by the other end-to-end neural architectures. Using the pre-trained Wiki embeddings helps improving the accuracy of the fastText method in all cases. Among the approaches with the SVM classifier, Wiki+SVM performs best in the single-source settings, but shows lower accuracy than Handcr.+SVM in the multi-source (TRG–ALL) settings. Wiki+Gauss.+SVM performs worst apart from on ES–EN and DE–ALL.

In the monolingual single-source settings, we observe that accuracy is slightly lower when the source language is typologically closer to the text language, i.e. it becomes more difficult to detect translationese. Specifically, DE–EN tends to have lower accuracy than DE–ES; EN–DE lower accuracy than EN–ES; and ES–EN lower accuracy than ES–DE. Accuracy generally drops when going from single-source to the multi-source setting, e.g. from DE–EN and DE–ES to DE–ALL. The EN–ALL dataset is the most difficult for most of the models among the TRG–ALL datasets. The ALL–ALL[3] setting exhibits comparable accuracy to the TRG–ALL setting for the neural models, but for the SVM there is a drop of around 9 points. Throughout our discussion we always report absolute differences between systems. The ALL–ALL[8] data results in reduced accuracy for most architectures, except Handcr.+SVM.

Neural-classifier-based models substantially outperform the other architectures: the SVMs trained with hand-crafted linguistically-inspired features,

|  | DE–EN | DE–ES | EN–DE | EN–ES | ES–DE | ES–EN | DE–ALL | EN–ALL | ES–ALL |
|---|---|---|---|---|---|---|---|---|---|
| DE–EN | 92.4±0.2 | 76.6±0.7 | - | - | - | - | 90.5±0.3 | - | - |
| DE–ES | 82.6±1.1 | 94.4±0.1 | - | - | - | - | 91.8±0.4 | - | - |
| EN–DE | - | - | 90.7±0.1 | 64.7±1.4 | - | - | - | 87.3±0.4 | - |
| EN–ES | - | - | 72.9±0.9 | 91.9±0.4 | - | - | - | 88.6±0.4 | - |
| ES–DE | - | - | - | - | 92.3±0.2 | 78.8±0.9 | - | - | 90.6±0.1 |
| ES–EN | - | - | - | - | 78.8±1.6 | 91.4±0.3 | - | - | 89.0±0.2 |
| DE–ALL | 87.3±0.6 | 85.3±0.4 | - | - | - | - | 90.9±0.3 | - | - |
| EN–ALL | - | - | 81.7±0.5 | 78.3±0.7 | - | - | - | 87.9±0.4 | - |
| ES–ALL | - | - | - | - | 85.9±0.9 | 85.0±0.6 | - | - | 89.9±0.1 |

Table 3: BERT translationese classification accuracy of all TRG–SRC and TRG–ALL models on TRG–SRC and TRG–ALL test sets (average and standard deviation over 5 runs). Columns: training set; rows: test set.

e.g., trail BERT by ∼20 accuracy points.

To make sure our hand-crafted-feature-based SVM results are competitive, we compare them with Rabinovich and Wintner (2015) on our data. Rabinovich and Wintner (2015) show that training a SVM classifier on the top 1000 most frequent POS- or character-trigrams yields SOTA translationese classification results on Europarl data. On our data, POS-trigrams yield around 5 points increase in accuracy for most of the datasets and character-trigrams tend to lower the accuracy by around 4 points (Appendix A.3). For the remainder of the paper we continue to work with our hand-crafted features, designed to capture various linguistic aspects of translationese.

## 5 Multilinguality and Cross-Language Performance

Since neural architectures perform better than the non-neural ones, we perform the multilingual and cross-language analysis only with the neural models. We evaluate the models trained on one dataset on the other ones, in order to verify:

- Whether for a given target language, the model trained to detect translationese from one source language, can detect translationese from another source language: TRG–SRC$_1$ on TRG–SRC$_2$, and TRG–SRC on TRG–ALL;
- How well the model trained to detect translationese from multiple source languages can detect translationese from a single source language: TRG–ALL on TRG–SRC, and ALL–ALL[3] on TRG–SRC;
- How well the model trained to detect translationese in multilingual data performs on monolingual data: ALL–ALL[3] on TRG–ALL, and ALL–ALL[3] on TRG–SRC.

Table 3 shows the results of cross-data testing for the monolingual models for the best-performing architecture (BERT). For the single-source monolingual models, we observe a relatively smaller drop (up to 13 percentage points) in performance when testing TRG–SRC on TRG–ALL (as compared to testing TRG–SRC on TRG–SRC), and a larger drop (up to 27 points) when testing TRG–SRC$_1$ on TRG–SRC$_2$ (as compared to testing TRG–SRC$_1$ on TRG–SRC$_2$). The fact that classification performance stays above 64% confirms the hypothesis that translationese features are source-language-independent.

Another trend that can be observed is that in cross testing TRG–SRC$_1$ and TRG–SRC$_2$, the model where the source language is more distant from the target suffers larger performance drop when tested on the test set with the closer-related source language, than the other way around. For instance, the DE–ES model tested on the DE–EN data suffers a decrease of 17.8 points, and DE–EN model tested on the DE–ES data suffers a decrease of 9.8 points. This may be due to DE–EN having learned more of the general translationese features, which helps the model to obtain higher accuracy on the data with a different source, while the DE–ES model may have learned to rely more on the language-pair-specific features, and therefore it gives lower accuracy on the data with the different source. A similar observation has been made by Koppel and Ordan (2011).

For the multi-source monolingual models (TRG–ALL), testing on TRG–SRC$_1$ and TRG–SRC$_2$ datasets shows a slight increase in performance for a source language that is more distant from the target, and a slight decrease for the more closely-related source language (as compared to testing TRG–ALL on TRG–ALL).

Table 4 displays the results of testing the

| | Handcr. | fastText | Simpl. | LSTM | BERT |
|---|---|---|---|---|---|
| | +SVM | (FT) | Trf. | | |
| DE–EN | 58.5±0.0 (↓13.0) | 85.9±0.0 (↓2.5) | 85.5±0.5 (↓4.3) | 86.6±0.7 (↓2.9) | 90.5±0.3 (↓1.9) |
| DE–ES | 57.0±0.0 (↓19.2) | 88.3±0.0 (↓2.6) | 87.2±0.5 (↓4.3) | 85.3±0.3 (↓6.9) | 91.5±0.2 (↓2.9) |
| EN–DE | 50.0±0.0 (↓17.6) | 81.5±0.1 (↓3.6) | 81.1±0.3 (↓4.7) | 80.9±0.3 (↓5.8) | 87.2±0.4 (↓3.5) |
| EN–ES | 50.5±0.0 (↓19.6) | 84.6±0.0 (↓3.0) | 83.5±0.5 (↓5.9) | 83.8±0.6 (↓5.3) | 88.9±0.3 (↓3.0) |
| ES–DE | 50.0±0.0 (↓21.0) | 85.6±0.0 (↓2.8) | 86.2±0.4 (↓4.3) | 85.7±0.5 (↓4.6) | 90.4±0.4 (↓1.9) |
| ES–EN | 51.3±0.0 (↓15.4) | 84.1±0.0 (↓2.9) | 84.6±0.4 (↓0.4) | 82.1±0.4 (↓6.7) | 89.0±0.4 (↓2.4) |
| DE–ALL | 59.9±0.0 (↓12.7) | 87.2±0.0 (↓0.2) | 86.3±0.4 (↓2.3) | 85.9±0.5 (↓2.6) | 90.8±0.1 (↓0.1) |
| EN–ALL | 50.2±0.0 (↓15.1) | 82.9±0.0 (↑0.2) | 82.0±0.1 (↓1.8) | 82.2±0.2 (↓2.1) | 88.1±0.5 (↑0.2) |
| ES–ALL | 50.0±0.0 (↓17.4) | 84.8±0.0 (↓0.1) | 85.3±0.2 (↓1.6) | 85.2±0.5 (↓1.8) | 89.8±0.3 (↓0.1) |
| ALL–ALL[3] | 58.9±0.0 (0.0) | 85.0±0.0 (0.0) | 84.5±0.2 (0.0) | 84.4±0.3 (0.0) | 89.6±0.2 (0.0) |

Table 4: Translationese classification accuracy of the ALL–ALL[3] model on all test sets (average and standard deviations over 5 runs). The difference from actual trained model performance is indicated in parentheses.

| | Handcr. | fastText | Simpl. | LSTM | BERT |
|---|---|---|---|---|---|
| | +SVM | (FT) | Trf. | | |
| DE–EN | 53.0±0.5 (↓18,5) | 71.0±0.3 (↓17.4) | 79.3±0.4 (↓12.3) | 79.9±0.5 (↓9.6) | 85.5±0.4 (↓6.9) |
| DE–ES | 51.3±0.3 (↓24.9) | 73.2±0.3 (↓17.7) | 81.4±0.3 (↓8.4) | 79.0±0.5 (↓12.9) | 87.9±0.3 (↓6.5) |
| EN–DE | 48.3±0.1 (↓19.3) | 65.8±0.2 (↓19.3) | 74.2±1.0 (↓11.6) | 72.9±0.4 (↓13.8) | 79.0±0.5 (↓11.7) |
| EN–ES | 50.3±0.1 (↓19.8) | 68.9±0.3 (↓18.7) | 76.8±0.6 (↓12.8) | 75.6±0.8 (↓13.5) | 83.2±0.4 (↓8.7) |
| ES–DE | 50.0±0.0 (↓21.0) | 71.1±0.2 (↓17.3) | 78.8±0.5 (↓11.6) | 76.0±0.7 (↓14.2) | 83.8±0.3 (↓8.5) |
| ES–EN | 53.2±0.5 (↓13.5) | 69.9±0.2 (↓17.1) | 76.7±0.6 (↓11.7) | 75.4±0.7 (↓13.4) | 82.8±0.2 (↓8.6) |
| DE–ALL | 53.1±0.5 (↓19.5) | 72.1±0.3 (↓15.3) | 80.5±0.4 (↓8.1) | 79.7±0.6 (↓8.8) | 86.8±0.2 (↓4.1) |
| EN–ALL | 48.4±0.2 (↓16.9) | 67.0±0.2 (↓15.7) | 75.4±0.9 (↓8.4) | 74.4±0.5 (↓9.9) | 81.1±0.1 (↓6.8) |
| ES–ALL | 50.8±0.3 (↓16.6) | 70.4±0.2 (↓14.5) | 77.9±0.6 (↓9.1) | 75.9±0.6 (↓11.1) | 83.2±0.3 (↓6.7) |
| ALL–ALL[3] | 53.2±0.3 (↓5.7) | 70.5±0.2 (↓14.5) | 77.9±0.2 (↓6.6) | 76.7±0.5 (↓7.7) | 83.7±0.1 (↓5.9) |
| ALL–ALL[8] | 65.4±0.1 (0.0) | 70.4±0.1 (0.0) | 77.9±0.1 (0.0) | 77.2±0.3 (0.0) | 84.6±0.2 (0.0) |

Table 5: As Table 4 for the ALL–ALL[8] model.

multilingual (ALL–ALL[3]) models on all test sets for the neural architectures, as well as Handcr.+SVM. We observe that the largest performance drop (as compared to testing on ALL–ALL[3] test set) happens for the EN–DE test set. For the DE–ES set, the performance actually increases for the neural models, but not for the Handcr.+SVM. We extended this experiment in Table 5, testing the ALL–ALL[8] on all test sets to further complement our multilingual analysis with more diverse languages and observe a similar trend, which is in line with the accuracy of the ALL–ALL[3] models on all test sets.

We also compare the performance of ALL–ALL[3] on different test sets to the original performance of the models trained on these datasets (in parentheses). There is a relatively larger drop in accuracy for the TRG–SRC data, than for TRG–ALL data. The largest drop for neural models is 6.7 accuracy points whilst the smallest performance drop for the Handcr.+SVM is 12.7. This highlights the ability of the neural models to learn features in a multilingual setting which generalize well to their component languages whereas the Handcr.+SVM method does not seem to work well for such a case. However, for ALL–ALL[8] models, Table 5 shows a large performance drop across all architectures as compared to the results from the models specifically trained for the task. The actual models are trained on language-specific features, whereas the ALL–ALL[8] model is trained on more diverse data containing typologically distant languages and thus captures less targeted translationese signals.

In summary, we observe that:

- For a given target language, even though a neural model trained on one source language can decently identify translationese from another source language, the decrease in performance is substantial.
- Neural models trained on multiple sources for a given target language perform reasonably well on single-source languages.
- Neural models trained on multilingual data ALL–ALL[3] perform reasonably well on monolingual data, especially for multi-source monolingual data.

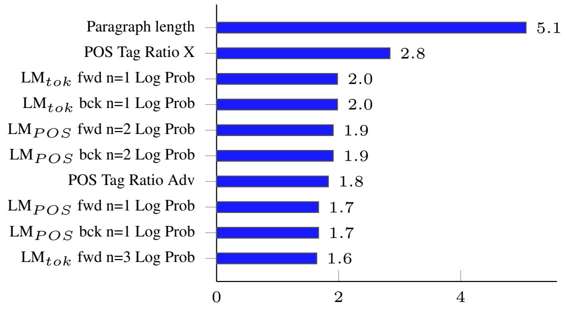


Figure 1: Top 10 SVM features, as a function of the absolute value of its feature weight.

- Using more source and target languages (ALL–ALL[8]) leads to a larger decrease in cross-testing accuracy.

## 6 Feature Importance and Relation to Neural Models

In this section we aim to quantify the feature importance of the hand-crafted linguistically inspired features used in `Handcr.+SVM` according to different multilingual models (ALL–ALL[3] setting).

As we use a Support Vector Machine with a linear kernel, we can interpret the magnitude of the feature weights as a feature importance measure. Guyon et al. (2002) for instance, use the squared SVM weights for feature selection. We rank the features by the absolute value of the weight. The feature ranks are listed in Appendix A.1. Figure 1 shows the top 10 features. Paragraph length is the most relevant feature, and we observe that most of the top features correspond to paragraph log probability. These features characterize simplification in translationese.

To explore whether there is any correlation between the hand-crafted features and predictions of the trained neural models, we conduct the following experiment in the multilingual setting. We fit a linear regression model for each hand-crafted feature, using the estimated probabilities of neural model as gold labels to be predicted. More formally, with $n$ paragraphs ($\mathrm{p}_i$, $i = 1...n$) in the test set and $d$ features, for each feature vector $\mathbf{x}_j \in R^n$, $j = 1...d$ we fit a model

$$\mathbf{y} = w_j \mathbf{x}_j + b_j, \tag{3}$$

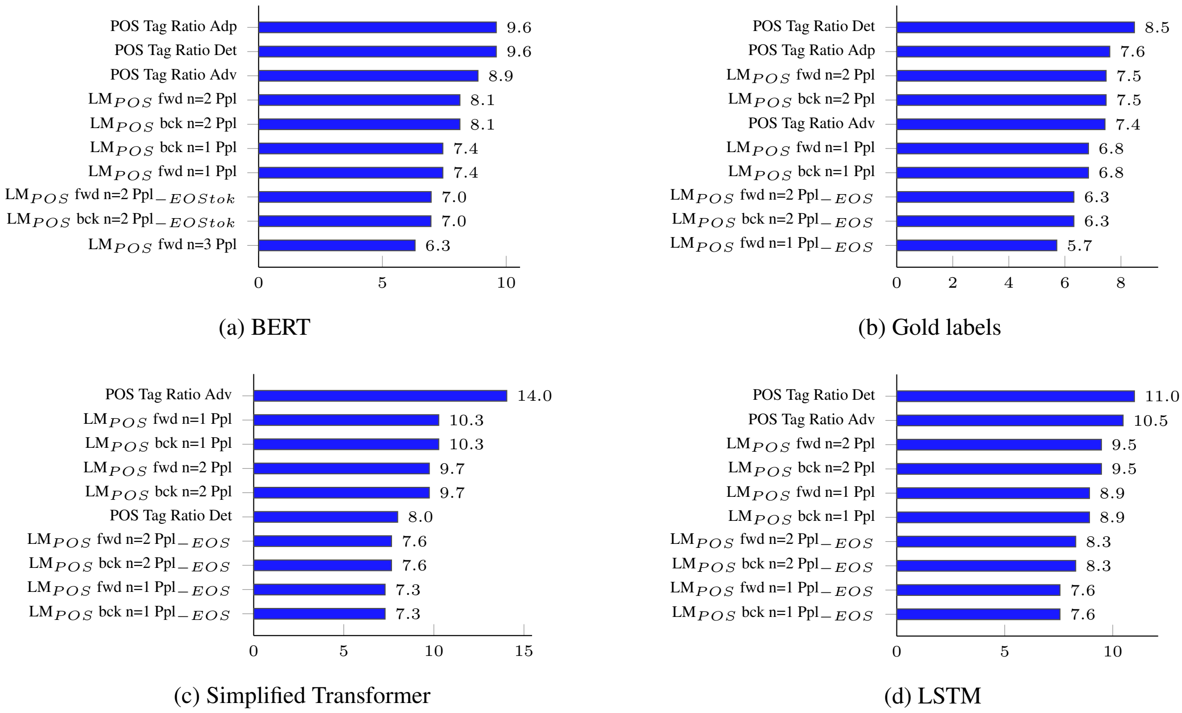
where $w_j, b_j \in R$ are the model parameters, and $\mathbf{y} \in R^n$ is a vector of predictions of the neural model $F$ (LSTM, Simplified Transformer, BERT) on the test set, with each dimension $y_i$ showing the probability of a data point to belong to the translationese class:

$$\mathbf{y}_i = \mathrm{P}(F(\mathrm{p}_i) = 1) \tag{4}$$

We apply min–max normalization to the features. We find that a large proportion of the linguistically motivated features are statistically significant for predicting the neural models' predicted probabilities, namely 60 features (out of 108) are significant for LSTM, 38 for the Simplified Transformer, and 56 for BERT, each with probability $99.9\%$. We also fit the per-feature linear models to predict the actual gold labels (and not the predictions of the neural models) to investigate which features correlate with the ground truth classes, and find 55 features to be statistically significant with $99.9\%$ probability. The full list of statistically significant features for each model, as well as for the gold labels is given in the Appendix A.1. We observe that the features significant for the neural models largely overlap with the features significant for the gold labels: the $F_1$-score (as a measure of overlap) is 0.89 for LSTM, 0.75 for Simplified Transformer and 0.99 for BERT. This is expected, because high-performing neural models output probabilities that are generally close to the gold labels, therefore a similar correlation with hand-crafted features occurs.

The $R^2$ measure is further used to rank features based on the amount of explained variance in predictions of a model. The top 10 features for predicting the predictions of each neural model and for predicting the actual gold labels are displayed in Figure 2. The order of top features is similar across the neural models (pairwise rank correlations $\rho_{Spearman}$ of at least 0.76), and similar to, but not identical to, the gold label results (pairwise rank correlations $\rho_{Spearman}$ of at least 0.75). We observe that most of the top features are either POS-perplexity-based, or bag-of-POS features. These features characterize interference in translationese. It also appears that more importance is attached to perplexities based on unigrams and bigrams than on other $n$-grams. Notably, the order of feature importance for the neural models is highly dissimilar from the order of hand-crafted feature weights for the SVM (pairwise rank correlations $\rho_{Spearman}$ at most 0.23). This might be connected to an accuracy gap between these models.

We conclude that many of hand-crafted translationese features are statistically significant for



(a) BERT

(b) Gold labels

(c) Simplified Transformer

(d) LSTM

Figure 2: Top 10 features as a function of $R^2 \cdot 10^{-3}$ for the neural architectures and the gold labels.

predicting the predictions of the neural models (and actual gold labels). However, due to the low $R^2$ values, we cannot conclude that the hand-crafted features explain the features learnt by the representation-learning models.

## 7 Summary and Conclusion

This paper presents a systematic comparison of the performance of feature-engineering-based and feature-learning-based models on binary translationese classification tasks in various settings, i.e., monolingual single-source data, monolingual multi-source data, and multilingual multi-source data. Additionally, we analyze neural architectures to see how well the hand-crafted features explain the variance in the predictions of neural models. The results obtained in our experiments show that, $(i)$ representation-learning-based approaches outperform hand-crafted linguistically inspired feature-selection methods for translationese classification on a wide range of tasks, $(ii)$ the features learned by feature-learning based methods generalise better to different multilingual tasks and $(iii)$ our multilingual experiments provide empirical support for the existence of language independent translationese features. We also examine multiple neural architectures and confirm that translationese classification requires deep neural models for optimum results. We have shown that many traditional hand-crafted translationese features significantly predict the output of representation learning models, but may not necessarily explain their performance due to the weak correlation. Our experiments also show that even though single-source monolingual models yield the best performance, they can, to a reasonable extent, be substituted by multi-source mono- and multi-lingual models.

Our interpretability experiment provides only some initial insight into the neural models' performance. Even though there are significant relationships between many of the features and the neural models' predicted probabilities, further experiments are required to verify that the neural models actually use something akin to these features. Also our current approach ignores interaction between the features. In the future, we plan to conduct a more detailed analysis of the neural models' decision making.

## Acknowledgements

We would like to thank the reviewers for their insightful comments and feedback. This research is funded by the German Research Foundation (Deutsche Forschungsgemeinschaft) under grant SFB 1102: Information Density and Linguistic Encoding.

## References

Kwabena Amponsah-Kaakyire, Daria Pylypenko, Cristina España-Bonet, and Josef van Genabith. 2021. Do not rely on relay translations: Multilingual parallel direct europarl. In *Workshop on Modelling Translation: Translatology in the Digital Age*. 23rd Nordic Conference on Computational Linguistics.

Ehud Alexander Avner, Noam Ordan, and Shuly Wintner. 2016. Identifying translationese at the word and sub-word level. *Digit. Scholarsh. Humanit.*, 31:30–54.

Mona Baker, Gill Francis, and Elena Tognini-Bonelli. 1993. Corpus linguistics and translation studies: Implications and applications. In *Text and Technology: In Honour of John Sinclair*, page 233–, Netherlands. John Benjamins Publishing Company.

Marco Baroni and Silvia Bernardini. 2005. A New Approach to the Study of Translationese: Machine-learning the Difference between Original and Translated Text. *Literary and Linguistic Computing*, 21(3):259–274.

Yuri Bizzoni, Tom S Juzek, Cristina España-Bonet, Koel Dutta Chowdhury, Josef van Genabith, and Elke Teich. 2020. How Human is Machine Translationese? Comparing Human and Machine Translations of Text and Speech. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 280–290. Association for Computational Linguistics.

Johannes Bjerva, Robert Östling, Maria Han Veiga, Jörg Tiedemann, and Isabelle Augenstein. 2019. What do language representations really represent? *Computational Linguistics*, 45(2):381–389.

Nikolay Bogoychev and Rico Sennrich. 2019. Domain, translationese and noise in synthetic data for neural machine translation. *CoRR*, abs/1911.03362.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 795–804.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Koel Dutta Chowdhury, Cristina España-Bonet, and Josef van Genabith. 2020. Understanding translationese in multi-view embedding spaces. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6056–6062.

Koel Dutta Chowdhury, Cristina España-Bonet, and Josef van Genabith. 2021. Tracing source language interference in translation with graph-isomorphism measures. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2021*, pages 380–390, Varna, Bulgaria.

Sergey Edunov, Myle Ott, Marc'Aurelio Ranzato, and Michael Auli. 2020. On the evaluation of machine translation systems trained with back-translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2836–2846, Online. Association for Computational Linguistics.

Markus Freitag, Isaac Caswell, and Scott Roy. 2019. APE at scale and its implications on MT evaluation biases. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 34–44, Florence, Italy. Association for Computational Linguistics.

Martin Gellerstam. 1986. Translationese in Swedish novels translated from English. *Translation studies in Scandinavia*, 1:88–95.

Antoine Gourru, Julien Velcin, and Julien Jacques. 2020. Gaussian embedding of linked documents from a pretrained semantic space. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3912–3918. Main track.

Yvette Graham, Barry Haddow, and Philipp Koehn. 2020. Statistical power and translationese in machine translation evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 72–81, Online. Association for Computational Linguistics.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Iustina Ilisei, Diana Inkpen, Gloria Corpas Pastor, and Ruslan Mitkov. 2010. Identification of translationese: A machine learning approach. In *Computational Linguistics and Intelligent Text Processing*, pages 503–511, Berlin, Heidelberg. Springer Berlin Heidelberg.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

D. Kenny. 2001. *Lexis and Creativity in Translation: A Corpus-based Study*. St. Jerome Pub.

Moshe Koppel and Noam Ordan. 2011. Translationese and its dialects. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1318–1326, Portland, Oregon, USA. Association for Computational Linguistics.

David Kurokawa, Cyril Goutte, and P. Isabelle. 2009. Automatic detection of translated text and its impact on machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, Maryland.

Sara Laviosa. 1998. Core patterns of lexical use in a comparable corpus of english narrative prose. *Meta: Journal des traducteurs*, 43.

Giannis Nikolentzos, Polykarpos Meladianos, François Rousseau, Yannis Stavrakas, and Michalis Vazirgiannis. 2017. Multivariate gaussian document representation from word embeddings for text categorization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 450–455.

Sergiu Nisioi, Ella Rabinovich, Liviu P. Dinu, and Shuly Wintner. 2016. A corpus of native, non-native and translated texts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4197–4201, Portorož, Slovenia. European Language Resources Association (ELRA).

Marius Popescu. 2011. Studying translationese at the character level. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 634–639, Hissar, Bulgaria. Association for Computational Linguistics.

Ella Rabinovich, Noam Ordan, and Shuly Wintner. 2017. Found in translation: Reconstructing phylogenetic language trees from translations. *arXiv preprint arXiv:1704.07146*.

Ella Rabinovich and Shuly Wintner. 2015. Unsupervised identification of translationese. *Transactions of the Association for Computational Linguistics*, 3:419–432.

Ella Rabinovich, Shuly Wintner, and Ofek Luis Lewinsohn. 2015. The Haifa Corpus of Translationese. *CoRR*, abs/1509.03611.

Parker Riley, Isaac Caswell, Markus Freitag, and David Grangier. 2020. Translationese as a language in "multilingual" NMT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7737–7746, Online. Association for Computational Linguistics.

Raphael Rubino, Ekaterina Lapshinova-Koltunski, and Josef van Genabith. 2016. Information Density and Quality Estimation Features as Translationese Indicators for Human Translation Classification. In *Proceedings of NAACL-HLT 2016, Association for Computational Linguistics*, pages 960–970, San Diego, California.

Ilia Sominsky and Shuly Wintner. 2019. Automatic detection of translation direction. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1131–1140, Varna, Bulgaria. INCOMA Ltd.

Andreas Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002)*, pages 901–904.

Sara Stymne. 2017. The effect of translationese on tuning for statistical machine translation. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 241–246, Gothenburg, Sweden. Association for Computational Linguistics.

Ahmad Taie, Raphael Rubino, and Josef van Genabith. 2018. INFODENS: an open-source framework for learning text representations. *CoRR*, abs/1810.07091.

Elad Tolochinsky, Ohad Mosafi, Ella Rabinovich, and Shuly Wintner. 2018. The UN parallel corpus annotated for translation direction. *CoRR*, abs/1805.07697.

Antonio Toral, Sheila Castilho, Ke Hu, and Andy Way. 2018. Attaining the unattainable? reassessing claims of human parity in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 113–123, Brussels, Belgium. Association for Computational Linguistics.

Gideon Toury. 1980. *In Search of a Theory of Translation*. The Porter Institute for Poetics and Semiotics, Tel Aviv University, Tel Aviv.

Hans van Halteren. 2008. Source language markers in EUROPARL translations. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 937–944, Manchester, UK. Coling 2008 Organizing Committee.

Vered Volansky, Noam Ordan, and Shuly Wintner. 2015. On the features of translationese. *Digital Scholarship in the Humanities*, 30(1):98–118.

Mike Zhang and Antonio Toral. 2019. The effect of translationese in machine translation test sets. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 73–81, Florence, Italy. Association for Computational Linguistics.

# A Appendix

## A.1 List of hand-crafted features

**Col. 3:** SVM feature importance ranks (ranked by absolute feature weight) for the model trained on the ALL–ALL[3] set.

**Col. 4-7:** Statistical significance of the features as predictors in per-feature linear regression with respect to neural models' predicted probabilities and gold labels (1 –significant with 99.9% confidence level) on the ALL–ALL[3] test set.

| ID | Feature name | SVM rank | Simplified Transformer | LSTM | BERT | Gold labels |
|---|---|---|---|---|---|---|
| **Surface features** | | | | | | |
| 0 | Average word length | 11 | 1 | 1 | 1 | 1 |
| 1 | Syllable ratio | 54 | 1 | 1 | 1 | 1 |
| 2 | Paragraph length | 1 | 0 | 1 | 1 | 1 |
| **Lexical features** | | | | | | |
| 3 | Lexical density | 19 | 0 | 1 | 1 | 1 |
| 4 | Type-token ratio | 16 | 0 | 0 | 0 | 0 |
| **Unigram bag-of-POS** | | | | | | |
| 5 | POS Tag Ratio Adj | 78 | 0 | 1 | 1 | 1 |
| 6 | POS Tag Ratio Adp | 40 | 1 | 1 | 1 | 1 |
| 7 | POS Tag Ratio Adv | 7 | 1 | 1 | 1 | 1 |
| 8 | POS Tag Ratio Aux | 83 | 0 | 0 | 0 | 0 |
| 9 | POS Tag Ratio Cconj | 80 | 0 | 0 | 0 | 0 |
| 10 | POS Tag Ratio Det | 18 | 1 | 1 | 1 | 1 |
| 11 | POS Tag Ratio Intj | 76 | 0 | 0 | 1 | 1 |
| 12 | POS Tag Ratio Noun | 15 | 1 | 1 | 1 | 1 |
| 13 | POS Tag Ratio Num | 46 | 0 | 0 | 0 | 0 |
| 14 | POS Tag Ratio Part | 89 | 0 | 0 | 0 | 0 |
| 15 | POS Tag Ratio Pron | 55 | 1 | 0 | 0 | 0 |
| 16 | POS Tag Ratio Propn | 26 | 0 | 1 | 1 | 1 |
| 17 | POS Tag Ratio Punct | 53 | 1 | 0 | 0 | 0 |
| 18 | POS Tag Ratio Sconj | 41 | 0 | 0 | 1 | 0 |
| 19 | POS Tag Ratio Space | 88 | 0 | 0 | 0 | 0 |
| 20 | POS Tag Ratio Sym | 66 | 0 | 0 | 0 | 0 |
| 21 | POS Tag Ratio Verb | 24 | 0 | 1 | 1 | 1 |
| 22 | POS Tag Ratio X | 2 | 1 | 1 | 1 | 1 |
| **SRILM language modelling features** | | | | | | |
| 23 | LM$_{tok}$ fwd n=1 Log Prob | 3 | 0 | 1 | 1 | 1 |
| 24 | LM$_{tok}$ fwd n=1 Ppl | 47 | 1 | 1 | 1 | 1 |
| 25 | LM$_{tok}$ fwd n=1 Ppl$_{-EOS}$ | 85 | 1 | 1 | 1 | 1 |
| 26 | LM$_{tok}$ fwd n=2 Log Prob | 20 | 0 | 1 | 1 | 1 |
| 27 | LM$_{tok}$ fwd n=2 Ppl | 45 | 0 | 0 | 0 | 0 |
| 28 | LM$_{tok}$ fwd n=2 Ppl$_{-EOS}$ | 96 | 0 | 0 | 0 | 0 |
| 29 | LM$_{tok}$ fwd n=3 Log Prob | 10 | 0 | 1 | 0 | 0 |
| 30 | LM$_{tok}$ fwd n=3 Ppl | 28 | 0 | 0 | 0 | 0 |
| 31 | LM$_{tok}$ fwd n=3 Ppl$_{-EOS}$ | 101 | 0 | 0 | 0 | 0 |
| 32 | LM$_{tok}$ fwd n=4 Log Prob | 14 | 0 | 1 | 0 | 0 |

Continued on next page

**Table 6 – continued from previous page**

| ID | Feature name | SVM rank | Simplified Transformer | LSTM | BERT | Gold labels |
|---|---|---|---|---|---|---|
| 33 | LM$_{tok}$ fwd n=4 Ppl | 27 | 0 | 0 | 0 | 0 |
| 34 | LM$_{tok}$ fwd n=4 Ppl$_{-EOS}$ | 100 | 0 | 0 | 0 | 0 |
| 35 | LM$_{tok}$ fwd n=5 Log Prob | 57 | 0 | 1 | 0 | 0 |
| 36 | LM$_{tok}$ fwd n=5 Ppl | 29 | 0 | 0 | 0 | 0 |
| 37 | LM$_{tok}$ fwd n=5 Ppl$_{-EOS}$ | 102 | 0 | 0 | 0 | 0 |
| 38 | LM$_{tok}$ bck n=1 Log Prob | 4 | 0 | 1 | 1 | 1 |
| 39 | LM$_{tok}$ bck n=1 Ppl | 48 | 1 | 1 | 1 | 1 |
| 40 | LM$_{tok}$ bck n=1 Ppl$_{-EOS}$ | 86 | 1 | 1 | 1 | 1 |
| 41 | LM$_{tok}$ bck n=2 Log Prob | 17 | 0 | 1 | 1 | 1 |
| 42 | LM$_{tok}$ bck n=2 Ppl | 52 | 0 | 0 | 0 | 0 |
| 43 | LM$_{tok}$ bck n=2 Ppl$_{-EOS}$ | 95 | 0 | 0 | 0 | 0 |
| 44 | LM$_{tok}$ bck n=3 Log Prob | 25 | 0 | 1 | 0 | 0 |
| 45 | LM$_{tok}$ bck n=3 Ppl | 32 | 0 | 0 | 0 | 0 |
| 46 | LM$_{tok}$ bck n=3 Ppl$_{-EOS}$ | 98 | 0 | 0 | 0 | 0 |
| 47 | LM$_{tok}$ bck n=4 Log Prob | 30 | 0 | 1 | 0 | 0 |
| 48 | LM$_{tok}$ bck n=4 Ppl | 31 | 0 | 0 | 0 | 0 |
| 49 | LM$_{tok}$ bck n=4 Ppl$_{-EOS}$ | 97 | 0 | 0 | 0 | 0 |
| 50 | LM$_{tok}$ bck n=5 Log Prob | 61 | 0 | 1 | 0 | 0 |
| 51 | LM$_{tok}$ bck n=5 Ppl | 37 | 0 | 0 | 0 | 0 |
| 52 | LM$_{tok}$ bck n=5 Ppl$_{-EOS}$ | 99 | 0 | 0 | 0 | 0 |
| 53 | LM$_{POS}$ fwd n=1 Log Prob | 8 | 0 | 1 | 1 | 1 |
| 54 | LM$_{POS}$ fwd n=1 Ppl | 33 | 1 | 1 | 1 | 1 |
| 55 | LM$_{POS}$ fwd n=1 Ppl$_{-EOS}$ | 58 | 1 | 1 | 1 | 1 |
| 56 | LM$_{POS}$ fwd n=2 Log Prob | 5 | 0 | 1 | 0 | 0 |
| 57 | LM$_{POS}$ fwd n=2 Ppl | 22 | 1 | 1 | 1 | 1 |
| 58 | LM$_{POS}$ fwd n=2 Ppl$_{-EOS}$ | 93 | 1 | 1 | 1 | 1 |
| 59 | LM$_{POS}$ fwd n=3 Log Prob | 12 | 0 | 1 | 1 | 1 |
| 60 | LM$_{POS}$ fwd n=3 Ppl | 63 | 1 | 1 | 1 | 1 |
| 61 | LM$_{POS}$ fwd n=3 Ppl$_{-EOS}$ | 106 | 1 | 1 | 1 | 1 |
| 62 | LM$_{POS}$ fwd n=4 Log Prob | 43 | 0 | 1 | 1 | 1 |
| 63 | LM$_{POS}$ fwd n=4 Ppl | 38 | 1 | 1 | 1 | 1 |
| 64 | LM$_{POS}$ fwd n=4 Ppl$_{-EOS}$ | 104 | 1 | 1 | 1 | 1 |
| 65 | LM$_{POS}$ fwd n=5 Log Prob | 75 | 0 | 1 | 1 | 1 |
| 66 | LM$_{POS}$ fwd n=5 Ppl | 35 | 1 | 1 | 1 | 1 |
| 67 | LM$_{POS}$ fwd n=5 Ppl$_{-EOS}$ | 103 | 1 | 1 | 1 | 1 |
| 68 | LM$_{POS}$ bck n=1 Log Prob | 9 | 0 | 1 | 1 | 1 |
| 69 | LM$_{POS}$ bck n=1 Ppl | 34 | 1 | 1 | 1 | 1 |
| 70 | LM$_{POS}$ bck n=1 Ppl$_{-EOS}$ | 59 | 1 | 1 | 1 | 1 |
| 71 | LM$_{POS}$ bck n=2 Log Prob | 6 | 0 | 1 | 0 | 0 |
| 72 | LM$_{POS}$ bck n=2 Ppl | 23 | 1 | 1 | 1 | 1 |
| 73 | LM$_{POS}$ bck n=2 Ppl$_{-EOS}$ | 94 | 1 | 1 | 1 | 1 |
| 74 | LM$_{POS}$ bck n=3 Log Prob | 13 | 0 | 1 | 1 | 1 |
| 75 | LM$_{POS}$ bck n=3 Ppl | 68 | 1 | 1 | 1 | 1 |
| 76 | LM$_{POS}$ bck n=3 Ppl$_{-EOS}$ | 108 | 1 | 1 | 1 | 1 |
| 77 | LM$_{POS}$ bck n=4 Log Prob | 42 | 0 | 1 | 1 | 1 |
| 78 | LM$_{POS}$ bck n=4 Ppl | 51 | 1 | 1 | 1 | 1 |

Continued on next page

**Table 6 – continued from previous page**

| ID | Feature name | SVM rank | Simplified Transformer | LSTM | BERT | Gold labels |
|---|---|---|---|---|---|---|
| 79 | LM$_{POS}$ bck n=4 Ppl$_{-EOS}$ | 107 | 1 | 1 | 1 | 1 |
| 80 | LM$_{POS}$ bck n=5 Log Prob | 67 | 0 | 1 | 1 | 1 |
| 81 | LM$_{POS}$ bck n=5 Ppl | 44 | 1 | 1 | 1 | 1 |
| 82 | LM$_{POS}$ bck n=5 Ppl$_{-EOS}$ | 105 | 1 | 1 | 1 | 1 |
| **N-gram freq. quartile distribution features** | | | | | | |
| 83 | % unigrams from freq. quartile 1 | 50 | 0 | 0 | 1 | 1 |
| 84 | % unigrams from freq. quartile 2 | 39 | 1 | 1 | 1 | 1 |
| 85 | % unigrams from freq. quartile 3 | 65 | 0 | 0 | 0 | 0 |
| 86 | % unigrams from freq. quartile 4 | 90 | 1 | 1 | 1 | 1 |
| 87 | % OOV unigrams | 21 | 1 | 1 | 1 | 1 |
| 88 | % bigrams from freq. quartile 1 | 36 | 1 | 1 | 0 | 0 |
| 89 | % bigrams from freq. quartile 2 | 79 | 1 | 0 | 1 | 1 |
| 90 | % bigrams from freq. quartile 3 | 70 | 0 | 0 | 0 | 0 |
| 91 | % bigrams from freq. quartile 4 | 60 | 0 | 0 | 0 | 0 |
| 92 | % OOV bigrams | 69 | 0 | 0 | 0 | 0 |
| 93 | % trigrams from freq. quartile 1 | 77 | 0 | 0 | 0 | 0 |
| 94 | % trigrams from freq. quartile 2 | 49 | 0 | 0 | 0 | 0 |
| 95 | % trigrams from freq. quartile 3 | 56 | 0 | 0 | 0 | 0 |
| 96 | % trigrams from freq. quartile 4 | 64 | 0 | 0 | 0 | 0 |
| 97 | % OOV trigrams | 81 | 0 | 0 | 0 | 0 |
| 98 | % 4-grams from freq. quartile 1 | 82 | 0 | 0 | 1 | 1 |
| 99 | % 4-grams from freq. quartile 2 | 73 | 0 | 0 | 0 | 0 |
| 100 | % 4-grams from freq. quartile 3 | 72 | 0 | 0 | 0 | 0 |
| 101 | % 4-grams from freq. quartile 4 | 74 | 0 | 0 | 0 | 0 |
| 102 | % OOV 4-grams | 62 | 0 | 0 | 0 | 0 |
| 103 | % 5-grams from freq. quartile 1 | 84 | 0 | 0 | 0 | 0 |
| 104 | % 5-grams from freq. quartile 2 | 87 | 0 | 0 | 0 | 0 |
| 105 | % 5-grams from freq. quartile 3 | 91 | 0 | 0 | 0 | 0 |
| 106 | % 5-grams from freq. quartile 4 | 92 | 0 | 0 | 0 | 0 |
| 107 | % OOV 5-grams | 71 | 0 | 0 | 0 | 0 |

### A.2 Simplified Transformer

The simplified transformer differs from the standard transformer in the following ways:

1. A cumulative sum-based contextualisation layer is used instead of positional encodings.
2. The attention computation is reduced to element-wise operations and has no feedforward connections.

#### A.2.1 Encoder

The encoder consists a contextualisation and attention layer with residual connections between both sublayers followed by layer normalisation.

#### A.2.2 Contextualisation

Given an input sequence $S = s_1, s_2, ..., s_L$, we obtain an embeddings matrix $X \in \mathbb{R}^{D \times L}$. The embeddings matrix $X$ is fed into the contextualisation layer of the transformer to obtain contextual embeddings $\hat{X} \in \mathbb{R}^{D \times L}$. We begin by taking a cumulative sum of the sequence of the embeddings $X$ as the context matrix

$$C = \sum_{j=1}^{L} X_{:j} \tag{5}$$

followed by a column-wise dot product between the embeddings matrix ($X$) and the generated context ($C$) to get weights $w \in \mathbb{R}^{1 \times L}$:

$$w_j = X_j \cdot C_j \tag{6}$$

where $j$ is the position of a word in the sequence $X$, $w$ is a row vector and $w_j$ is a scalar at position $j$. The original embeddings matrix ($X$) is then multiplied element-wise with the weights $w$ to obtain a contextualised representation of the sequence ($\hat{X}$):

$$\hat{X} = X_{ij} \cdot w_j \tag{7}$$

#### A.2.3 Attention

The attention takes 3 inputs: query, key and value. The output of the contextualisation layer ($\hat{X}$) is fed in as both the query and value. The context matrix ($C$) is fed in as the key. The query and key are passed through a feature map to obtain Q and K respectively. The feature map (8) ensures Q and K are always positive. Therefore we can simplify softmax to the first term in the product in (13): the Energy simply scaled by the sum of the Energies. The attention computation is formalised as follows:

$$Feature\ map(x) = gelu(x) + 1 \tag{8}$$

$$Q = |Feature\ map(query)| \tag{9}$$

$$K = |Feature\ map(key)| \tag{10}$$

$$V = value \tag{11}$$

$$Energy(E) = \frac{Q_{ij} \cdot K_{ij}}{\sqrt{D}} \tag{12}$$

$$Attention(A) = \frac{E_j}{\sum_{j=1}^{L} E_j} \cdot V \tag{13}$$

#### A.2.4 Decoder

The decoder consists of two blocks. The first block is similar to the encoder block with a contextualisation layer and attention layer with residual connections between both sublayers followed by layer normalisation. The second block is another attention layer with residual connection to the previous block followed by a layer normalisation. In the second block, the output of the first block is fed in as the query and the output of the encoder block is fed in as the value. The key is the sum of the encoder's embedding matrix ($X$). This sum operation can be skipped by taking the last column of the encoder's context matrix ($C$). The decoder output ($Y \in \mathbb{R}^{D \times L}$) is pooled (average) and the resulting $D \times 1$ vector is passed to a classifier.

### A.3 Handcrafted Feature-based SVM Baseline

For the POS-trigrams and character-trigrams baselines we implement the setup from Rabinovich and Wintner (2015) (and, respectively, Volansky et al. (2015)). In both cases we only take 1000 most frequent trigrams. The values correspond to the relative frequency of the trigram in the paragraph normalized by the total number of trigrams in the paragraph. For POS-trigrams, we pad the paragraphs with special start-of-string and end-of-string tokens. For character trigrams, we pad each word in this way, and avoid cross-token trigrams, as well as punctuation. Results are displayed in Table 7.

Rabinovich and Wintner (2015) report much higher accuracy (> 90) on their Europarl data: they classify text chunks of 2000 tokens, while we report results on paragraphs with average length of around 80 tokens.

| | POS-trigrams | char-trigrams |
|---|---|---|
| DE–EN | 76.6±0.0 (↑5.1) | 67.5±0.0 (↓4.1) |
| DE–ES | 80.3±0.0 (↑4.1) | 71.2±0.0 (↓5.0) |
| EN–DE | 73.1±0.0 (↑5.5) | 62.9±0.0 (↓4.8) |
| EN–ES | 73.4±0.0 (↑3.4) | 66.1±0.0 (↓4.0) |
| ES–DE | 76.1±0.0 (↑5.1) | 66.6±0.0 (↓4.4) |
| ES–EN | 74.2±0.0 (↑7.6) | 64.4±0.0 (↓2.3) |
| DE–ALL | 76.7±0.0 (↑4.1) | 68.4±0.0 (↓4.2) |
| EN–ALL | 69.7±0.0 (↑4.4) | 62.1±0.0 (↓3.3) |
| ES–ALL | 72.9±0.0 (↑5.5) | 63.3±0.0 (↓4.1) |
| ALL–ALL[3] | 66.6±0.0 (↑7.6) | 62.2±0.1 (↑3.3) |
| ALL–ALL[8] | 61.6±0.0 (↓3.8) | 61.5±0.0 (↓3.9) |

Table 7: Test accuracy of baseline systems implemented from (Rabinovich and Wintner, 2015). The mean and the standard deviations over 5 runs are reported. The difference from the Handcr.+SVM model is indicated in parentheses.