Information Content Measures of Semantic Similarity Perform Better Without Sense-Tagged Text

Ted Pedersen
Department of Computer Science
University of Minnesota, Duluth
Duluth, MN 55812
tpederse@d.umn.edu
http://wn-similarity.sourceforge.net

Abstract

This paper presents an empirical comparison of similarity measures for pairs of concepts based on Information Content. It shows that using modest amounts of untagged text to derive Information Content results in higher correlation with human similarity judgments than using the largest available corpus of manually annotated sense–tagged text.

1 Introduction

Measures of semantic similarity based on WordNet have been widely used in Natural Language Processing. These measures rely on the structure of WordNet to produce a numeric score that quantifies the degree to which two concepts (represented by a sense or synset) are similar (or not). In their simplest form these measures use path length to identify concepts that are physically close to each other and therefore considered to be more similar than concepts that are further apart.

While this is a reasonable first approximation to semantic similarity, there are some well known limitations. Most significant is that path lengths between very specific concepts imply much smaller distinctions in semantic similarity than do comparable path lengths between very general concepts. One proposed improvement is to augment concepts in WordNet with *Information Content* values derived from sense–tagged corpora or from raw unannotated corpora (Resnik, 1995).

This paper shows that Information Content measures based on modest amounts of unannotated corpora have greater correlation with human similarity judgements than do those based on the largest corpus of sense-tagged text currently available.[1] The key to this success is not in the specific type of corpora used, but rather in increasing the number of concepts in WordNet that have counts associated with them. These results show that Information Content measures of semantic similarity can be significantly improved without requiring the creation of sense–tagged corpora (which is very expensive).

1.1 Information Content

Information Content (IC) is a measure of specificity for a concept. Higher values are associated with more specific concepts (e.g., *pitch_fork*), while those with lower values are more general (e.g., *idea*). Information Content is computed based on frequency counts of concepts as found in a corpus of text. The frequency associated with a concept is incremented in WordNet each time that concept is observed, as are the counts of the ancestor concepts in the WordNet hierarchy (for nouns and verbs). This is necessary because each occurrence of a more specific concept also implies the occurrence of the more general ancestor concepts.

When a corpus is sense–tagged, mapping occurrences of a word to a concept is straightforward (since each sense of a word corresponds with a concept or synset in WordNet). However, if the text has not been sense–tagged then all of the possible senses of a given word are incremented (as are their ancestors). For example, if *tree* (as a plant) occurs in a sense–tagged text, then only the concept associated

[1] These experiments were done with version 2.05 of WordNet::Similarity (Pedersen et al., 2004).

Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, pages 329–332,
Los Angeles, California, June 2010.

with tree as a kind of plant would be incremented. If the text is untagged, then all of the possible senses of *tree* would be incremented (such as the mathematical sense of tree, a shoe tree, a plant, etc.) In this case the frequency of all the occurrences of a word are divided equally among the different possible senses. Thus, if a word occurs 42 times in a corpus and there are six possible senses (concepts), each sense and all of their ancestors would have their frequency incremented by seven.[2]

For each concept (synset) c in WordNet, Information Content is defined as the negative log of the probability of that concept (based on the observed frequency counts):

$$IC(c) = -logP(c)$$

Information Content can only be computed for nouns and verbs in WordNet, since these are the only parts of speech where concepts are organized in hierarchies. Since these hierarchies are separate, Information Content measures of similarity can only be applied to pairs of nouns or pairs of verbs.

2 Semantic Similarity Measures

There are three Information Content measures implemented in WordNet::Similarity: (res) (Resnik, 1995), (jcn) (Jiang and Conrath, 1997), and (lin) (Lin, 1998).

These measures take as input two concepts c_1 and c_2 (i.e., senses or synsets in WordNet) and output a numeric measure of similarity. These measures all rely to varying degrees on the idea of a least common subsumer (LCS); this is the most specific concept that is a shared ancestor of the two concepts. For example, the LCS of *automobile* and *scooter* is *vehicle*.

The Resnik (res) measure simply uses the Information Content of the LCS as the similarity value:

$$res(c_1, c_2) = IC(LCS(c_1, c_2))$$

The Resnik measure is considered somewhat coarse, since many different pairs of concepts may share the same LCS. However, it is less likely to suffer from zero counts (and resulting undefined values) since in general the LCS of two concepts will not be a very specific concept (i.e., a leaf node in WordNet), but will instead be a somewhat more general concept that is more likely to have observed counts associated with it.

Both the Lin and Jiang & Conrath measures attempt to refine the Resnik measure by augmenting it with the Information Content of the individual concepts being measured in two different ways:

$$lin(c_1, c_2) = \frac{2*res(c_1,c_2)}{IC(c_1)+IC(c_2)}$$

$$jcn(c_1, c_2) = \frac{1}{IC(c_1)+IC(c_2)-2*res(c_1,c_2)}$$

All three of these measures have been widely used in the NLP literature, and have tended to perform well in a wide range of applications such as word sense disambiguation, paraphrase detection, and Question Answering (c.f., (Resnik, 1999)).

3 Experimental Data

Information Content in WordNet::Similarity is (by default) derived from SemCor (Miller et al., 1993), a manually sense–tagged subset of the Brown Corpus. It is made up of approximately 676,000 words, of which 226,000 are sense–tagged. SemCor was originally created using sense–tags from version 1.6 of WordNet, and has been mapped to subsequent versions to stay current.[3] This paper uses version 3.0 of WordNet and SemCor.

WordNet::Similarity also includes a utility (rawtextFreq.pl) that allows a user to derive Information Content values from any corpus of plain text. This utility is used with the untagged version of SemCor and with various portions of the English GigaWord corpus (1st edition) to derive alternative Information Content values.

English GigaWord contains more than 1.7 billion words of newspaper text from the 1990's and early 21st century, divided among four different sources: Agence France Press English Service (afe), Associated Press Worldstream English Service (apw), The New York Times Newswire Service (nyt), and The Xinhua News Agency English Service (xie).

This paper compares the ranking of pairs of concepts according to Information Content measures in WordNet::Similarity with a number of manually created gold standards. These include the (RG) (Rubenstein and Goodenough, 1965) collection of 65 noun

[2]This is the –resnik counting option in WordNet::Similarity.

[3]http://www.cse.unt.edu/~rada/downloads.html

Table 1: Rank Correlation of Existing Measures

measure	WS	MC	RG
vector	**.46**	**.89**	**.73**
lesk	.42	.83	.68
wup	.34	.74	.69
lch	.28	.71	.70
path	.26	.68	.69
random	-.20	-.16	.15

pairs, the (MC) (Miller and Charles, 1991) collection of 30 noun pairs (a subset of RG), and the (WS) WordSimilarity-353 collection of 353 pairs (Finkelstein et al., 2002). RG and MC have been scored for similarity, while WS is scored for relatedness, which is a more general and less well–defined notion than similarity. For example *aspirin* and *headache* are clearly related, but they aren't really similar.

4 Experimental Results

Table 1 shows the Spearman's rank correlation of several other measures of similarity and relatedness in WordNet::Similarity with the gold standards discussed above. The WordNet::Similarity vector relatedness measure achieves the highest correlation, followed closely by the adapted lesk measure. These results are consistent with previous findings (Patwardhan and Pedersen, 2006). This table also shows results for several path–based measures.[4]

Table 2 shows the correlation of jcn, res, and lin when Information Content is derived from 1) the sense-tagged version of SemCor (semcor), 2) SemCor without sense tags (semcor-raw), and 3) steadily increasing subsets of the 133 million word xie portion of the English GigaWord corpus. These subsets start with the entire first month of xie (199501, from January 1995) and then two months (199501-02), three months (199501-03), up through all of 1995 (199501-12). Thereafter the increments are annual, with two years of data (1995-1996), then three (1995-1997), and so on until the entire xie corpus is used (1995-2001). The afe, apw, and nyt portions of GigaWord are also used individually and then combined all together along with xie *(all)*.

[4] *wup* is the Wu & Palmer measure, *lch* is the Leacock & Chodorow measure, *path* relies on edge counting, and *random* provides a simple sanity check.

The size (in tokens) of each corpus is shown in the second column of Table 2 *(size)*, which is expressed in thousands (k), millions (m), and billions (b).

The third column (*cover*) shows what percentage of the 96,000 noun and verb synsets in WordNet receive a non-zero frequency count when Information Content is derived from the specified corpus. These values show that the 226,000 sense–tagged instances in SemCor cover about 24%, and the untagged version of SemCor covers 37%. As it happens the correlation results for semcor-raw are somewhat better than semcor, suggesting that coverage is at least as important (if not more so) to the performance of Information Content measures than accurate mapping of words to concepts.

A similar pattern can be seen with the xie results in Table 2. This again shows that an increase in WordNet coverage is associated with increased performance of the Information Content measures. As coverage increases the correlation improves, and in fact the results are better than the path–based measures and approach those of lesk and vector (see Table 1). The one exception is with respect to the WS gold standard, where vector and lesk perform much better than the Information Content measures. However, this seems reasonable since they are relatedness measures, and the WS corpus is annotated for relatedness rather than similarity.

As a final test of the hypothesis that coverage matters as much or more than accurate mapping of words to concepts, a simple baseline method was created that assigns each synset a count of 1, and then propagates that count up to the ancestor concepts. This is equivalent to doing add-1 smoothing without any text (add1only). This results in correlation nearly as high as the best results with xie and semcor-raw, and is significantly better than semcor.

5 Conclusions

This paper shows that semantic similarity measures based on Information Content can be significantly improved by increasing the coverage of the frequency counts used to derive Information Content. Increased coverage can come from unannotated text or simply assigning counts to every concept in WordNet and does not require sense–tagged text.

Table 2: Rank Correlation of Information Content Measures From Different Corpora

			jcn			lin			res		
corpus	size	cover	WS	MC	RG	WS	MC	RG	WS	MC	RG
semcor	226 k	.24	.21	.72	.51	.30	.73	.58	.38	.74	.69
semcor-raw	670 k	.37	.26	.82	.58	.32	.79	.65	.38	.76	.70
xie:											
199501	1.2 m	.35	.35	.78	.57	.37	.75	.63	.37	.73	.68
199501-02	2.3 m	.39	.31	.79	.65	.32	.75	.67	.36	.73	.68
199501-03	3.8 m	.42	.34	.88	.69	.34	.81	.70	.37	.75	.69
199501-06	7.9 m	.46	.36	.88	.69	.36	.81	.70	.37	.75	.69
199501-09	12 m	.49	.36	.88	.69	.36	.81	.70	.37	.75	.69
199501-12	16 m	.51	.37	.87	.73	.36	.81	.71	.37	.75	.69
1995-1996	34 m	.56	.37	.88	.73	.36	.81	.72	.37	.75	.69
1995-1997	53 m	.58	.37	.88	.73	.36	.81	.71	.37	.75	.69
1995-1998	73 m	.60	.37	.89	.73	.36	.81	.72	.37	.75	.69
1995-1999	94 m	.62	.36	.88	.73	.36	.81	.72	.37	.76	.69
1995-2000	115 m	.63	.36	.89	.73	.36	.81	.71	.37	.76	.70
1995-2001	133 m	.64	.36	.88	.73	.36	.81	.71	.37	.76	.70
afe	174 m	.66	**.36**	**.88**	**.81**	.36	.80	.78	.37	.77	.79
apw	560 m	.75	.36	.84	.78	.36	.79	.78	.37	.76	.79
nyt	963 m	.83	.36	.84	.78	.36	.79	.77	.37	.77	.80
all	1.8 b	.85	.34	.85	.79	.35	.80	.78	.37	.77	.79
add1only	96 k	1.00	.36	.85	.73	.37	.77	.73	.39	.76	.70

Acknowledgements

Many thanks to Siddharth Patwardhan and Jason Michelizzi for their exceptional work on WordNet::Similarity over the years, which has made this and a great deal of other research possible.

References

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

J. Jiang and D. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings on International Conference on Research in Computational Linguistics*, pages 19–33, Taiwan.

D. Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the International Conference on Machine Learning*, Madison, August.

G.A. Miller and W.G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

G.A. Miller, C. Leacock, R. Tengi, and R. Bunker. 1993. A semantic concordance. In *Proceedings of the Workshop on Human Language Technology*, pages 303–308.

S. Patwardhan and T. Pedersen. 2006. Using WordNet-based Context Vectors to Estimate the Semantic Relatedness of Concepts. In *Proceedings of the EACL 2006 Workshop on Making Sense of Sense: Bringing Computational Linguistics and Psycholinguistics Together*, pages 1–8, Trento, Italy, April.

T. Pedersen, S. Patwardhan, and J. Michelizzi. 2004. Wordnet::Similarity - Measuring the relatedness of concepts. In *Proceedings of Fifth Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 38–41, Boston, MA.

P. Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, Montreal, August.

P. Resnik. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130.

H. Rubenstein and J.B. Goodenough. 1965. Contextual correlates of synonymy. *Computational Linguistics*, 8:627–633.

Linguistic Regularities in Continuous Space Word Representations

Tomas Mikolov*, **Wen-tau Yih, Geoffrey Zweig**
Microsoft Research
Redmond, WA 98052

Abstract

Continuous space language models have recently demonstrated outstanding results across a variety of tasks. In this paper, we examine the vector-space word representations that are implicitly learned by the input-layer weights. We find that these representations are surprisingly good at capturing syntactic and semantic regularities in language, and that each relationship is characterized by a relation-specific vector offset. This allows vector-oriented reasoning based on the offsets between words. For example, the male/female relationship is automatically learned, and with the induced vector representations, "King - Man + Woman" results in a vector very close to "Queen." We demonstrate that the word vectors capture syntactic regularities by means of syntactic analogy questions (provided with this paper), and are able to correctly answer almost 40% of the questions. We demonstrate that the word vectors capture semantic regularities by using the vector offset method to answer SemEval-2012 Task 2 questions. Remarkably, this method outperforms the best previous systems.

1 Introduction

A defining feature of neural network language models is their representation of words as high dimensional real valued vectors. In these models (Bengio et al., 2003; Schwenk, 2007; Mikolov et al., 2010), words are converted via a learned lookup-table into real valued vectors which are used as the inputs to a neural network. As pointed out by the original proposers, one of the main advantages of these models is that the distributed representation achieves a level of generalization that is not possible with classical n-gram language models; whereas a n-gram model works in terms of discrete units that have no inherent relationship to one another, a continuous space model works in terms of word vectors where similar words are likely to have similar vectors. Thus, when the model parameters are adjusted in response to a particular word or word-sequence, the improvements will carry over to occurrences of similar words and sequences.

By training a neural network language model, one obtains not just the model itself, but also the learned word representations, which may be used for other, potentially unrelated, tasks. This has been used to good effect, for example in (Collobert and Weston, 2008; Turian et al., 2010) where induced word representations are used with sophisticated classifiers to improve performance in many NLP tasks.

In this work, we find that the learned word representations in fact capture meaningful syntactic and semantic regularities in a very simple way. Specifically, the regularities are observed as constant vector offsets between pairs of words sharing a particular relationship. For example, if we denote the vector for word i as x_i, and focus on the singular/plural relation, we observe that $x_{apple} - x_{apples} \approx x_{car} - x_{cars}$, $x_{family} - x_{families} \approx x_{car} - x_{cars}$, and so on. Perhaps more surprisingly, we find that this is also the case for a variety of semantic relations, as measured by the SemEval 2012 task of measuring relation similarity.

*Currently at Google, Inc.

Proceedings of NAACL-HLT 2013, pages 746–751,
Atlanta, Georgia, 9–14 June 2013.

The remainder of this paper is organized as follows. In Section 2, we discuss related work; Section 3 describes the recurrent neural network language model we used to obtain word vectors; Section 4 discusses the test sets; Section 5 describes our proposed vector offset method; Section 6 summarizes our experiments, and we conclude in Section 7.

2 Related Work

Distributed word representations have a long history, with early proposals including (Hinton, 1986; Pollack, 1990; Elman, 1991; Deerwester et al., 1990). More recently, neural network language models have been proposed for the classical language modeling task of predicting a probability distribution over the "next" word, given some preceding words. These models were first studied in the context of feed-forward networks (Bengio et al., 2003; Bengio et al., 2006), and later in the context of recurrent neural network models (Mikolov et al., 2010; Mikolov et al., 2011b). This early work demonstrated outstanding performance in terms of word-prediction, but also the need for more computationally efficient models. This has been addressed by subsequent work using hierarchical prediction (Morin and Bengio, 2005; Mnih and Hinton, 2009; Le et al., 2011; Mikolov et al., 2011b; Mikolov et al., 2011a). Also of note, the use of distributed topic representations has been studied in (Hinton and Salakhutdinov, 2006; Hinton and Salakhutdinov, 2010), and (Bordes et al., 2012) presents a semantically driven method for obtaining word representations.

3 Recurrent Neural Network Model

The word representations we study are learned by a recurrent neural network language model (Mikolov et al., 2010), as illustrated in Figure 1. This architecture consists of an input layer, a hidden layer with recurrent connections, plus the corresponding weight matrices. The input vector $\mathbf{w}(t)$ represents input word at time t encoded using 1-of-N coding, and the output layer $\mathbf{y}(t)$ produces a probability distribution over words. The hidden layer $\mathbf{s}(t)$ maintains a representation of the sentence history. The input vector $\mathbf{w}(t)$ and the output vector $\mathbf{y}(t)$ have dimensionality of the vocabulary. The values in the hidden and output layers are computed as follows:

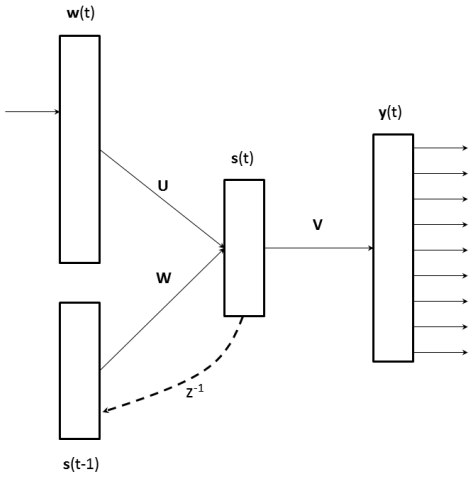


Figure 1: Recurrent Neural Network Language Model.

$$\mathbf{s}(t) = f\left(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t-1)\right) \quad (1)$$

$$\mathbf{y}(t) = g\left(\mathbf{V}\mathbf{s}(t)\right), \quad (2)$$

where

$$f(z) = \frac{1}{1+e^{-z}}, \quad g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}. \quad (3)$$

In this framework, the word representations are found in the columns of $\mathbf{U}$, with each column representing a word. The RNN is trained with backpropagation to maximize the data log-likelihood under the model. The model itself has no knowledge of syntax or morphology or semantics. Remarkably, training such a purely lexical model to maximize likelihood will induce word representations with striking syntactic and semantic properties.

4 Measuring Linguistic Regularity

4.1 A Syntactic Test Set

To understand better the syntactic regularities which are inherent in the learned representation, we created a test set of analogy questions of the form "*a* is to *b* as *c* is to ___" testing base/comparative/superlative forms of adjectives; singular/plural forms of common nouns; possessive/non-possessive forms of common nouns; and base, past and 3rd person present tense forms of verbs. More precisely, we tagged 267M words of newspaper text with Penn

Category	Relation	Patterns Tested	# Questions	Example
Adjectives	Base/Comparative	JJ/JJR, JJR/JJ	1000	good:better rough:___
Adjectives	Base/Superlative	JJ/JJS, JJS/JJ	1000	good:best rough:___
Adjectives	Comparative/ Superlative	JJS/JJR, JJR/JJS	1000	better:best rougher:___
Nouns	Singular/Plural	NN/NNS, NNS/NN	1000	year:years law:___
Nouns	Non-possessive/ Possessive	NN/NN_POS, NN_POS/NN	1000	city:city's bank:___
Verbs	Base/Past	VB/VBD, VBD/VB	1000	see:saw return:___
Verbs	Base/3rd Person Singular Present	VB/VBZ, VBZ/VB	1000	see:sees return:___
Verbs	Past/3rd Person Singular Present	VBD/VBZ, VBZ/VBD	1000	saw:sees returned:___

Table 1: Test set patterns. For a given pattern and word-pair, both orderings occur in the test set. For example, if "see:saw return:___" occurs, so will "saw:see returned:___".

Treebank POS tags (Marcus et al., 1993). We then selected 100 of the most frequent comparative adjectives (words labeled JJR); 100 of the most frequent plural nouns (NNS); 100 of the most frequent possessive nouns (NN_POS); and 100 of the most frequent base form verbs (VB). We then systematically generated analogy questions by randomly matching each of the 100 words with 5 other words from the same category, and creating variants as indicated in Table 1. The total test set size is 8000. The test set is available online. [1]

4.2 A Semantic Test Set

In addition to syntactic analogy questions, we used the SemEval-2012 Task 2, *Measuring Relation Similarity* (Jurgens et al., 2012), to estimate the extent to which RNNLM word vectors contain semantic information. The dataset contains 79 fine-grained word relations, where 10 are used for training and 69 testing. Each relation is exemplified by 3 or 4 gold word pairs. Given a group of word pairs that supposedly have the same relation, the task is to order the target pairs according to the *degree* to which this relation holds. This can be viewed as another analogy problem. For example, take the *Class-Inclusion:Singular_Collective* relation with the prototypical word pair *clothing:shirt*. To measure the degree that a target word pair *dish:bowl* has the same relation, we form the analogy "*clothing* is to *shirt* as *dish* is to *bowl*," and ask how valid it is.

[1] http://research.microsoft.com/en-us/projects/rnn/default.aspx

5 The Vector Offset Method

As we have seen, both the syntactic and semantic tasks have been formulated as analogy questions. We have found that a simple vector offset method based on cosine distance is remarkably effective in solving these questions. In this method, we assume relationships are present as vector offsets, so that in the embedding space, all pairs of words sharing a particular relation are related by the same constant offset. This is illustrated in Figure 2.

In this model, to answer the analogy question *a:b c:d* where d is unknown, we find the embedding vectors x_a, x_b, x_c (all normalized to unit norm), and compute $y = x_b - x_a + x_c$. y is the continuous space representation of the word we expect to be the best answer. Of course, no word might exist at that exact position, so we then search for the word whose embedding vector has the greatest cosine similarity to y and output it:

$$w^* = argmax_w \frac{x_w y}{\|x_w\|\|y\|}$$

When d is given, as in our semantic test set, we simply use $\cos(x_b - x_a + x_c, x_d)$ for the words

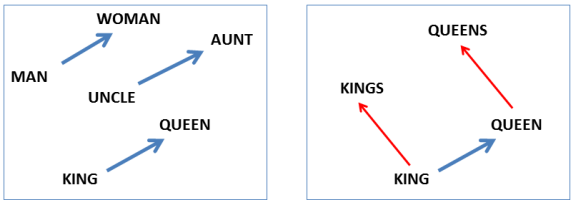


Figure 2: Left panel shows vector offsets for three word pairs illustrating the gender relation. Right panel shows a different projection, and the singular/plural relation for two words. In high-dimensional space, multiple relations can be embedded for a single word.

provided. We have explored several related methods and found that the proposed method performs well for both syntactic and semantic relations. We note that this measure is qualitatively similar to relational similarity model of (Turney, 2012), which predicts similarity between members of the word pairs $(x_b, x_d), (x_c, x_d)$ and dis-similarity for (x_a, x_d).

6 Experimental Results

To evaluate the vector offset method, we used vectors generated by the RNN toolkit of Mikolov (2012). Vectors of dimensionality 80, 320, and 640 were generated, along with a composite of several systems, with total dimensionality 1600. The systems were trained with 320M words of Broadcast News data as described in (Mikolov et al., 2011a), and had an 82k vocabulary. Table 2 shows results for both RNNLM and LSA vectors on the syntactic task. LSA was trained on the same data as the RNN. We see that the RNN vectors capture significantly more syntactic regularity than the LSA vectors, and do remarkably well in an absolute sense, answering more than one in three questions correctly. [2]

In Table 3 we compare the RNN vectors with those based on the methods of Collobert and Weston (2008) and Mnih and Hinton (2009), as implemented by (Turian et al., 2010) and available online [3] Since different words are present in these datasets, we computed the intersection of the vocabularies of the RNN vectors and the new vectors, and restricted the test set and word vectors to those. This resulted in a 36k word vocabulary, and a test set with 6632

[2]Guessing gets a small fraction of a percent.

[3]http://metaoptimize.com/projects/wordreprs/

Method	Adjectives	Nouns	Verbs	All
LSA-80	9.2	11.1	17.4	12.8
LSA-320	11.3	18.1	20.7	16.5
LSA-640	9.6	10.1	13.8	11.3
RNN-80	9.3	5.2	30.4	16.2
RNN-320	18.2	19.0	45.0	28.5
RNN-640	21.0	25.2	54.8	34.7
RNN-1600	**23.9**	**29.2**	**62.2**	**39.6**

Table 2: Results for identifying syntactic regularities for different word representations. Percent correct.

Method	Adjectives	Nouns	Verbs	All
RNN-80	**10.1**	8.1	**30.4**	**19.0**
CW-50	1.1	2.4	8.1	4.5
CW-100	1.3	4.1	8.6	5.0
HLBL-50	4.4	5.4	23.1	13.0
HLBL-100	7.6	**13.2**	30.2	18.7

Table 3: Comparison of RNN vectors with Turian's Collobert and Weston based vectors and the Hierarchical Log-Bilinear model of Mnih and Hinton. Percent correct.

questions. Turian's Collobert and Weston based vectors do poorly on this task, whereas the Hierarchical Log-Bilinear Model vectors of (Mnih and Hinton, 2009) do essentially as well as the RNN vectors. These representations were trained on 37M words of data and this may indicate a greater robustness of the HLBL method.

We conducted similar experiments with the semantic test set. For each target word pair in a relation category, the model measures its relational similarity to each of the prototypical word pairs, and then uses the average as the final score. The results are evaluated using the two standard metrics defined in the task, Spearman's rank correlation coefficient ρ and MaxDiff accuracy. In both cases, larger values are better. To compare to previous systems, we report the average over all 69 relations in the test set.

From Table 4, we see that as with the syntactic regularity study, the RNN-based representations perform best. In this case, however, Turian's CW vectors are comparable in performance to the HLBL vectors. With the RNN vectors, the performance improves as the number of dimensions increases. Surprisingly, we found that even though the RNN vec-

Method	Spearman's ρ	MaxDiff Acc.
LSA-640	0.149	0.364
RNN-80	0.211	0.389
RNN-320	0.259	0.408
RNN-640	0.270	0.416
RNN-1600	**0.275**	**0.418**
CW-50	0.159	0.363
CW-100	0.154	0.363
HLBL-50	0.149	0.363
HLBL-100	0.146	0.362
UTD-NB	0.230	0.395

Table 4: Results in measuring relation similarity

tors are not trained or tuned specifically for this task, the model achieves better results (RNN-320, RNN-640 & RNN-1600) than the previously best performing system, UTD-NB (Rink and Harabagiu, 2012).

7 Conclusion

We have presented a generally applicable vector offset method for identifying linguistic regularities in continuous space word representations. We have shown that the word representations learned by a RNNLM do an especially good job in capturing these regularities. We present a new dataset for measuring syntactic performance, and achieve almost 40% correct. We also evaluate semantic generalization on the SemEval 2012 task, and outperform the previous state-of-the-art. Surprisingly, both results are the byproducts of an unsupervised maximum likelihood training criterion that simply operates on a large amount of text data.

References

Y. Bengio, R. Ducharme, Vincent, P., and C. Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Reseach*, 3(6).

Y. Bengio, H. Schwenk, J.S. Senécal, F. Morin, and J.L. Gauvain. 2006. Neural probabilistic language models. *Innovations in Machine Learning*, pages 137–186.

A. Bordes, X. Glorot, J. Weston, and Y. Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *Proceedings of 15th International Conference on Artificial Intelligence and Statistics*.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(96).

J.L. Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2):195–225.

G.E. Hinton and R.R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

G. Hinton and R. Salakhutdinov. 2010. Discovering binary codes for documents by learning deep generative models. *Topics in Cognitive Science*, 3(1):74–91.

G.E. Hinton. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, pages 1–12. Amherst, MA.

David Jurgens, Saif Mohammad, Peter Turney, and Keith Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics (SemEval 2012)*, pages 356–364. Association for Computational Linguistics.

Hai-Son Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon. 2011. Structured output layer neural network language model. In *Proceedings of ICASSP 2011*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

Tomas Mikolov, Martin Karafiat, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech 2010*.

Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky. 2011a. Strategies for Training Large Scale Neural Network Language Models. In *Proceedings of ASRU 2011*.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2011b. Extensions of recurrent neural network based language model. In *Proceedings of ICASSP 2011*.

Tomas Mikolov. 2012. RNN toolkit.

A. Mnih and G.E. Hinton. 2009. A scalable hierarchical distributed language model. *Advances in neural information processing systems*, 21:1081–1088.

F. Morin and Y. Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the*

international workshop on artificial intelligence and statistics, pages 246–252.

J.B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1):77–105.

Bryan Rink and Sanda Harabagiu. 2012. UTD: Determining relational similarity using lexical patterns. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics (SemEval 2012)*, pages 413–418. Association for Computational Linguistics.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21(3):492 – 518.

J. Turian, L. Ratinov, and Y. Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of Association for Computational Linguistics (ACL 2010)*.

P.D. Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

The Microsoft Research Sentence Completion Challenge

Geoffry Zweig and Christopher J.C. Burges
Microsoft Research Technical Report MSR-TR-2011-129
February 20th, 2011

Abstract
Work on modeling semantics in text is progressing quickly, yet currently there are few public datasets which authors can use to measure and compare their systems. This work takes a step towards addressing this issue. We present the MSR Sentence Completion Challenge Data, which consists of 1,040 sentences, each of which has four *impostor* sentences, in which a single (fixed) word in the original sentence has been replaced by an impostor word with similar occurrence statistics. For each sentence the task is then to determine which of the five choices for that word is the correct one. This dataset was constructed from Project Gutenberg data. Seed sentences were selected from five of Sir Arthur Conan Doyle's Sherlock Holmes novels, and then imposter words were suggested with the aid of a language model trained on over 500 19th century novels. The language model was used to compute 30 alternative words for a given low frequency word in a sentence, and human judges then picked the 4 best impostor words, based on a set of provided guidelines. Although the data presented here will not be changed, this is still a work in progress, and we plan to add similar datasets based on other sources. This technical report is a living document and will be updated appropriately as new datasets are constructed and new results on existing datasets (for example, using human subjects) are reported.

1 Introduction

Interest in semantic modeling for text is growing rapidly (see for example [1, 2, 3, 4]). However, currently there are few publicly available large datasets with which researchers can compare results, and those that are available focus on isolated word pairs. For example, WordSimilarity-353 [5] consists of 353 word pairs whose degree of similarity has been determined by human judges. In [6], the authors make available a test set consisting of 950 questions in which the goal is to find the word that is most opposite in meaning to another.

Geoffrey Zweig and Christopher J.C. Burges
Microsoft Research, Redmond, WA., e-mail: {gzweig, chris.burges}@microsoft.com

As a step towards addressing this problem, we present a set of 1,040 English sentences, taken from five novels written by Sir Arthur Conan Doyle. Each sentence has associated with it four *impostor* sentences, in which a single (fixed) word in the original sentence has been replaced by an impostor word with similar occurrence statistics. For each sentence the task is then to determine which of the five choices for that word is the correct one. The task is thus similar to a language SAT test. Our dataset was constructed from 19th century novel data from Project Gutenberg. We chose to use this source because of the high quality of the English, and also to avoid any copyright issues. We chose to use a single author (Conan Doyle) for the target sentences to give a consistent style of writing. We plan to construct similar datasets in the future to help explore other axes (multiple authors, and modern English, such as is typical in Wikipedia). Our data can be found at *http://research.microsoft.com/scc/.*

2 The Question Generation Process

Question generation was done in two steps. First, a candidate sentence containing an infrequent word was selected, and alternates for that word were automatically determined by sampling with an n-gram language model. The n-gram model used the immediate history as context, thus resulting in words that make "look good" locally, but for which there is no a-priori reason to expect them to make sense globally. In the second step, we eliminated choices which are obviously incorrect because they constitute grammatical errors. Choices requiring semantic knowledge and logical inference were preferred, as described in the guidelines, which we give in section 3. Note that an important *desideratum* guiding the data generation process was requiring that a researcher who knows exactly how the data was created, including knowing which data was used to train the language model, should nevertheless not be able to use that information to solve the problem. We now describe the data that was used, and then describe the two steps in more detail.

2.1 Data Used

Seed sentences were selected from five of Conan Doyle's Sherlock Holmes novels: *The Sign of the Four (1890), The Hound of the Baskervilles (1892), The Adventures of Sherlock Holmes (1892), The Memoirs of Sherlock Holmes (1894),* and *The Valley of Fear (1915)*. Once a focus word within the sentence was selected, alternates to that word were generated using a n-gram language model. This model was trained on approximately 540 texts from the Project Gutenberg collection, consisting mainly of 19th century novels. Of these 522 had adequate headers attesting to lack of copyright, and they are now available the *Sentence Completion Challenge* website.

2.2 Automatically Generating Alternates

Alternates were generated for every sentence containing an infrequent word. A state-of-the-art class-based maximum entropy n-gram model [7] was used to generate the alternates. The following procedure was used:

1. Select a word with overall frequency less than 10^{-4}. For example, we might select "extraordinary" in "It is really the most extraordinary and inexplicable business."
2. Use the two-word history immediately preceding the selected focus word to predict alternates. We sampled 150 unique alternates at this stage, requiring that they all have frequency less than 10^{-4}. For example, "the most" predicts "handsome" and "luminous."
3. If the original (correct) sentence has a better score than any of these alternates, reject the sentence.
4. Else, score each option according to how well it and its immediate predecessor predict the next word. For example, the probability of "and" following "most handsome" might be 0.012.
5. Sort the predicted words according to this score, and retain the top 30 options.

In step 3, omitting questions for which the correct sentence is the best makes the set of options more difficult to solve with a langauge model alone. However, by allowing the correct sentence to potentially fall below the set of alternates retained, an opposite bias is created: the language model will tend to assign a lower score to the correct option than to the alternates (which were chosen by virtue of scoring well). We measured the bias by performing a test using the langauge model, and choosing the *lowest* scoring candidate as the answer. This gave an accuracy of 26% (as opposed to 31%, found by taking the highest scoring candidate). Thus although there is some remaining bias for the answer to be low scoring, it is small. When a language model other than the precise one used to generate the data is used, the score reversal test yielded 17% correct. The correct polarity gave 39%. If, however, just the single score used to do the sort in the last step is used (i.e. the probability of the immediate successor alone), then the lowest scoring alternate is correct about as much as the language model itself. Neither is anywhere close to human performance.

The overall procedure has the effect of providing options which are both well-predicted by the immediate history, and predictive of the immediate future. However, in total it uses just four consecutive words, and cannot be expected to provide globally coherent alternates.

2.3 Human Grooming

The human judges (who picked the best four choices of impostor sentences from the automatically generated list of thirty) were given the following instructions:

1. All chosen sentences should be grammatically correct. For example: *He dances while he ate his pipe* would be illegal.
2. Each correct answer should be unambiguous. In other words, the correct answer should always be a significantly better fit for that sentence than each of the four impostors; it should be possible to write down an explanation as to why the correct answer is the correct answer, that would persuade most reasonable people.
3. Sentences that might cause offense or controversy should be avoided.

4. Ideally the alternatives will require some thought in order to determine the correct answer. For example:

   - *Was she his [client | musings | discomfiture | choice | opportunity] , his friend , or his mistress?*

   would constitute a good test sentence. In order to arrive at the correct answer, the student must notice that, while *"musings"* and *"discomfiture"* are both clearly wrong, the terms *friend* and *mistress* both describe people, which therefore makes *client* a more likely choice than *choice* or *opportunity*.
5. Alternatives that require understanding properties of entities that are mentioned in the sentence are desirable. For example:

   - *All red-headed men who are above the age of [800 | seven | twenty-one | 1,200 | 60,000] years , are eligible.*

   requires that the student realize that a *man* cannot be seven years old, or 800 or more. However, such example are rare: most often, arriving at the answer will still require thought, but will not require detailed entity knowledge, such as:

   - *That is his [generous | mother's | successful | favorite | main] fault , but on the whole he's a good worker.*

6. We encourage the use of a dictionary, if necessary.
7. A given sentence should only occur once. If more than one target word has been identified for a sentence (i.e. different targets have been identified, in different positions), choose the set of sentences that generates the best challenge, according to the above guidelines.

Note that the impostors sometimes constitute a perfectly fine completion, but that in those cases, the correct completion is still clearly identifiable as the most likely completion.

3 Guidelines for Use

It is important for users of this data to realize the following: since the test data was taken from five 19th century novels, the test data itself is likely to occur in the index of most Web search engines, and in other large scale datasets that were constructed from web data (for example, the Google N-gram project). For example, entering the string *That is his fault , but on the whole he's a good worker* (one of the sentence examples given above, but with the target word removed) into the Bing search engine results in the correct (full) sentence at the top position. It is important to realize that researchers may inadvertently get better results than truly warranted because they have used data that is thus tainted by the test set. To help prevent any such criticism from being leveled at a particular publication, we recommend than in any set of published results, the exact data used for training and validation be specified.

4 Baseline Results

4.1 A Simple 4-gram model

As a sanity check we constructed a very simple N-gram model as follows: given a test sentence (with the position of the target word known), the score for that sentence was initialized to zero, and then incremented by one for each bigram match, by two for each trigram match, and by three for each 4-gram match, where a match means that the N-gram in the test sentence containing the target word occurs at least once in the background data. This simple method achieved 34% correct (compared to 20% by random choice) on the test set.

4.2 Smoothed N-gram model

As a somewhat more sophisticated baseline, we use the CMU language modeling toolkit [1] to build a 4-gram language model using Good-Turing smoothing. We kept all bigrams and trigrams occurring in the data, as well as four-grams occurring at least twice. We used a vocabulary of the 126k words that occurred five or more times, and this resulted in a total of 26M N-grams. This improved by 5% absolute on the simple baseline to achieve 39% correct.

4.3 Latent Semantic Analysis Similarity

As a final benchmark, we present scores for a novel method based on latent semantic analysis. In this approach, we treated each sentence in the training data as a "document" and performed latent semantic analysis [8] to obtain a 300 dimensional vector representation of each word in the vocabulary. Denoting two words by their vectors $\mathbf{x}, \mathbf{y}$, their similarity is defined as the cosine of the angle between them:

$$\mathtt{sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\| \mathbf{x} \| \| \mathbf{y} \|}.$$

To decide which option to select, we computed the average similarity to every other word in the sentence, and then output the word with the greatest overall similarity. This results in our best baseline performance, at 49% correct.

4.4 Benchmark Summary

Table 1 summarizes our benchmark study. First, for reference, we had an unaffiliated human answer a random subset of 100 questions. Ninety-one percent were answered correctly, showing that scores

[1] *http://www.speech.cs.cmu.edu/SLM/toolkit.html*

in the range of 90% are reasonable to expect. Secondly, we tested the performance of the same model (Model M) that was used to generate the data. Because this model output alternates that it assigns high-probability, there is a bias against it, and it scored 31%. Smoothed 3 and 4-gram models built with the CMU toolkit achieved 36 to 39 percent. The simple 4-gram model described earlier did slightly worse (hampered by a lack of smoothing), and the LSA similarity model did best with 49%. As a further check on this data, we have run the same tests on 203 sentence completion questions from a practice SAT exam and achieve similar results (Princeton Review, *11 Practice Tests for the SAT & PSAT*, 2011 Edition). To train language models for the SAT question task, we used 1.2 billion words of Los Angeles Times data taken from the years 1985 through 2002.

Method	**% Correct** (N=1040)
Human	91
Generating Model	31
Smoothed 3-gram	36
Smoothed 4-gram	39
Simple 4-gram	34
Average LSA Similarity	49

Table 1 Summary of Benchmarks

These results indicate that the "Holmes" sentence completion set is indeed a challenging problem, with a level of difficulty roughly comparable to that of SAT questions. Simple models based on N-gram statistics do quite poorly, and even a relatively sophisticated semantic-coherence model struggles to beat the 50% mark.

5 Conclusions and Future Work

We plan to add a similarly sized dataset based on Wikipedia, and also to present results found by asking human judges (who have only a non-electronic dictionary at hand) to perform the test. These human tests will be done in-house, since using M-Turk raises the problem that it is not clear how to construct the correct incentive (paying by the sentence alone will give poor accuracy, while paying by the correct sentence gives an incentive to taint the results by e.g. using a search engine). The in-house testing will also enable us to provide additional statistics regarding the judges' backgrounds, for example, their level of education, and whether or not they are native-born English speakers.

References

[1] Y. Bengio, R. Ducharme and P. Vincent. *A Neural Probabilistic Language Model.* Advances in Neural Information Processing Systems, 2001.

[2] R. Collobert and J. Weston and L. Bottou and M. Karlen and K. Kavukcuoglu and P.P. Kuksa. *Natural Language Processing (almost) from Scratch.* CoRR, http://arxiv.org/abs/1103.0398, 2011.

[3] R. Socher and J. Pennington and E.H. Huang and A.Y. Ng and C.D. Manning. *Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions.* EMNLP, 2011

[4] P. Blackburn and J. Bos. *Representation and Inference for Natural Language. A First Course in Computational Semantics* . CSLI Publications, 1999.
[5] Finkelstein, L. and Gabrilovich, Y.M. and Rivlin, E. and Solan, Z. and Wolfman, G. and Ruppin, E. *Placing search in context: The concept revisited.* ACM TOIS 20(1), 2002.
[6] Saif Mohammad, Bonnie Dorr , and Graeme Hirst. *Computing Word-Pair Antonymy* EMNLP, 2008.
[7] Stanley Chen. *Shrinking Exponential Language Models.* HLT 2009.
[8] Deerwester, S. and Dumais, S.T. and Furnas, G.W. and Landauer, T.K. and Harshman, R. *Indexing by Latent Semantic Analysis.* Journal of the American Society for Information Science, Vol. 41, 1990.

Appendix: Changelog

- Feb. 20, 2012 - Corrected the description of the sampling procedure in question generation. In Step 3, a sentence was discarded if the correct answer scored best (not worst) according to the language model prediction score.
- Feb. 20, 2012 - Replaced the list of training data with a downloadable set.

Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Character-Aware Neural Language Models

Yoon Kim
School of Engineering
and Applied Sciences
Harvard University
yoonkim@seas.harvard.edu

Yacine Jernite
Courant Institute
of Mathematical Sciences
New York University
jernite@cs.nyu.edu

David Sontag
Courant Institute
of Mathematical Sciences
New York University
dsontag@cs.nyu.edu

Alexander M. Rush
School of Engineering
and Applied Sciences
Harvard University
srush@seas.harvard.edu

Abstract

We describe a simple neural language model that relies only on character-level inputs. Predictions are still made at the word-level. Our model employs a convolutional neural network (CNN) and a highway network over characters, whose output is given to a long short-term memory (LSTM) recurrent neural network language model (RNN-LM). On the English Penn Treebank the model is on par with the existing state-of-the-art despite having 60% fewer parameters. On languages with rich morphology (Arabic, Czech, French, German, Spanish, Russian), the model outperforms word-level/morpheme-level LSTM baselines, again with fewer parameters. The results suggest that on many languages, character inputs are sufficient for language modeling. Analysis of word representations obtained from the character composition part of the model reveals that the model is able to encode, from characters only, both semantic and orthographic information.

Introduction

Language modeling is a fundamental task in artificial intelligence and natural language processing (NLP), with applications in speech recognition, text generation, and machine translation. A language model is formalized as a probability distribution over a sequence of strings (words), and traditional methods usually involve making an n-th order Markov assumption and estimating n-gram probabilities via counting and subsequent smoothing (Chen and Goodman 1998). The count-based models are simple to train, but probabilities of rare n-grams can be poorly estimated due to data sparsity (despite smoothing techniques).

Neural Language Models (NLM) address the n-gram data sparsity issue through parameterization of words as vectors (word embeddings) and using them as inputs to a neural network (Bengio, Ducharme, and Vincent 2003; Mikolov et al. 2010). The parameters are learned as part of the training process. Word embeddings obtained through NLMs exhibit the property whereby semantically close words are likewise close in the induced vector space (as is the case with non-neural techniques such as Latent Semantic Analysis (Deerwester, Dumais, and Harshman 1990)).

While NLMs have been shown to outperform count-based n-gram language models (Mikolov et al. 2011), they are blind to subword information (e.g. morphemes). For example, they do not know, a priori, that *eventful*, *eventfully*, *uneventful*, and *uneventfully* should have structurally related embeddings in the vector space. Embeddings of rare words can thus be poorly estimated, leading to high perplexities for rare words (and words surrounding them). This is especially problematic in morphologically rich languages with long-tailed frequency distributions or domains with dynamic vocabularies (e.g. social media).

In this work, we propose a language model that leverages subword information through a character-level convolutional neural network (CNN), whose output is used as an input to a recurrent neural network language model (RNN-LM). Unlike previous works that utilize subword information via morphemes (Botha and Blunsom 2014; Luong, Socher, and Manning 2013), our model does not require morphological tagging as a pre-processing step. And, unlike the recent line of work which combines input word embeddings with features from a character-level model (dos Santos and Zadrozny 2014; dos Santos and Guimaraes 2015), our model does not utilize word embeddings at all in the input layer. Given that most of the parameters in NLMs are from the word embeddings, the proposed model has significantly fewer parameters than previous NLMs, making it attractive for applications where model size may be an issue (e.g. cell phones).

To summarize, our contributions are as follows:

- on English, we achieve results on par with the existing state-of-the-art on the Penn Treebank (PTB), despite having approximately 60% fewer parameters, and
- on morphologically rich languages (Arabic, Czech, French, German, Spanish, and Russian), our model outperforms various baselines (Kneser-Ney, word-level/morpheme-level LSTM), again with fewer parameters.

We have released all the code for the models described in this paper.[1]

[1] https://github.com/yoonkim/lstm-char-cnn

Model

The architecture of our model, shown in Figure 1, is straightforward. Whereas a conventional NLM takes word embeddings as inputs, our model instead takes the output from a single-layer character-level convolutional neural network with max-over-time pooling.

For notation, we denote vectors with bold lower-case (e.g. $\mathbf{x}_t, \mathbf{b}$), matrices with bold upper-case (e.g. $\mathbf{W}, \mathbf{U}^o$), scalars with italic lower-case (e.g. x, b), and sets with cursive upper-case (e.g. $\mathcal{V}, \mathcal{C}$) letters. For notational convenience we assume that words and characters have already been converted into indices.

Recurrent Neural Network

A recurrent neural network (RNN) is a type of neural network architecture particularly suited for modeling sequential phenomena. At each time step t, an RNN takes the input vector $\mathbf{x}_t \in \mathbb{R}^n$ and the hidden state vector $\mathbf{h}_{t-1} \in \mathbb{R}^m$ and produces the next hidden state $\mathbf{h}_t$ by applying the following recursive operation:

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}) \tag{1}$$

Here $\mathbf{W} \in \mathbb{R}^{m \times n}, \mathbf{U} \in \mathbb{R}^{m \times m}, \mathbf{b} \in \mathbb{R}^m$ are parameters of an affine transformation and f is an element-wise nonlinearity. In theory the RNN can summarize all historical information up to time t with the hidden state $\mathbf{h}_t$. In practice however, learning long-range dependencies with a vanilla RNN is difficult due to vanishing/exploding gradients (Bengio, Simard, and Frasconi 1994), which occurs as a result of the Jacobian's multiplicativity with respect to time.

Long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) addresses the problem of learning long range dependencies by augmenting the RNN with a memory cell vector $\mathbf{c}_t \in \mathbb{R}^n$ at each time step. Concretely, one step of an LSTM takes as input $\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}$ and produces $\mathbf{h}_t$, $\mathbf{c}_t$ via the following intermediate calculations:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^i\mathbf{x}_t + \mathbf{U}^i\mathbf{h}_{t-1} + \mathbf{b}^i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}^f\mathbf{x}_t + \mathbf{U}^f\mathbf{h}_{t-1} + \mathbf{b}^f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}^o\mathbf{x}_t + \mathbf{U}^o\mathbf{h}_{t-1} + \mathbf{b}^o) \\
\mathbf{g}_t &= \tanh(\mathbf{W}^g\mathbf{x}_t + \mathbf{U}^g\mathbf{h}_{t-1} + \mathbf{b}^g) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{2}$$

Here $\sigma(\cdot)$ and $\tanh(\cdot)$ are the element-wise sigmoid and hyperbolic tangent functions, $\odot$ is the element-wise multiplication operator, and $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are referred to as *input*, *forget*, and *output* gates. At $t = 1$, $\mathbf{h}_0$ and $\mathbf{c}_0$ are initialized to zero vectors. Parameters of the LSTM are $\mathbf{W}^j, \mathbf{U}^j, \mathbf{b}^j$ for $j \in \{i, f, o, g\}$.

Memory cells in the LSTM are additive with respect to time, alleviating the gradient vanishing problem. Gradient exploding is still an issue, though in practice simple optimization strategies (such as gradient clipping) work well. LSTMs have been shown to outperform vanilla RNNs on many tasks, including on language modeling (Sundermeyer, Schluter, and Ney 2012). It is easy to extend the RNN/LSTM to two (or more) layers by having another network whose input at t is $\mathbf{h}_t$ (from the first network). Indeed, having multiple layers is often crucial for obtaining competitive performance on various tasks (Pascanu et al. 2013).

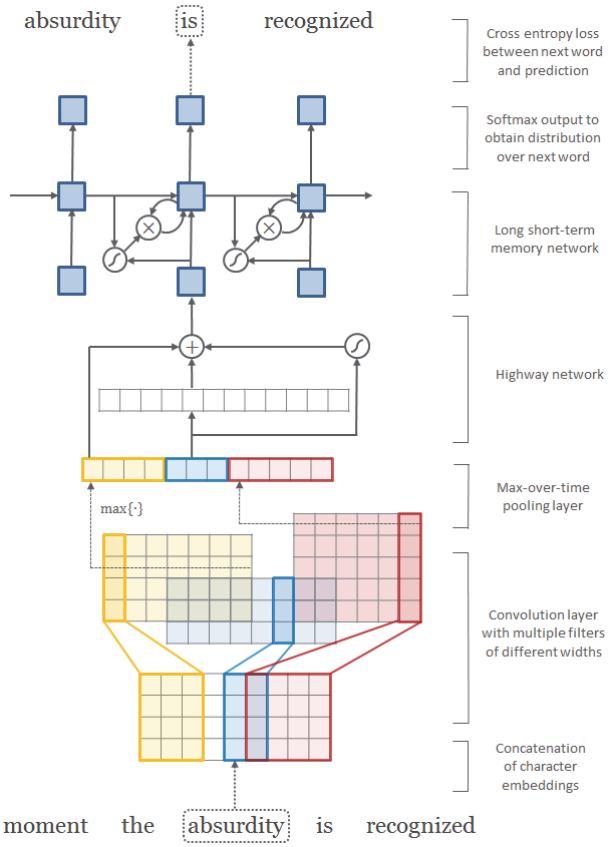


Figure 1: Architecture of our language model applied to an example sentence. Best viewed in color. Here the model takes *absurdity* as the current input and combines it with the history (as represented by the hidden state) to predict the next word, *is*. First layer performs a lookup of character embeddings (of dimension four) and stacks them to form the matrix $\mathbf{C}^k$. Then convolution operations are applied between $\mathbf{C}^k$ and multiple filter matrices. Note that in the above example we have twelve filters—three filters of width two (blue), four filters of width three (yellow), and five filters of width four (red). A max-over-time pooling operation is applied to obtain a fixed-dimensional representation of the word, which is given to the highway network. The highway network's output is used as the input to a multi-layer LSTM. Finally, an affine transformation followed by a softmax is applied over the hidden representation of the LSTM to obtain the distribution over the next word. Cross entropy loss between the (predicted) distribution over next word and the actual next word is minimized. Element-wise addition, multiplication, and sigmoid operators are depicted in circles, and affine transformations (plus nonlinearities where appropriate) are represented by solid arrows.

Recurrent Neural Network Language Model

Let $\mathcal{V}$ be the fixed size vocabulary of words. A language model specifies a distribution over w_{t+1} (whose support is $\mathcal{V}$) given the historical sequence $w_{1:t} = [w_1, \ldots, w_t]$. A recurrent neural network language model (RNN-LM) does this by applying an affine transformation to the hidden layer followed by a softmax:

$$\Pr(w_{t+1} = j | w_{1:t}) = \frac{\exp(\mathbf{h}_t \cdot \mathbf{p}^j + q^j)}{\sum_{j' \in \mathcal{V}} \exp(\mathbf{h}_t \cdot \mathbf{p}^{j'} + q^{j'})} \quad (3)$$

where $\mathbf{p}^j$ is the j-th column of $\mathbf{P} \in \mathbb{R}^{m \times |\mathcal{V}|}$ (also referred to as the *output embedding*),[2] and q^j is a bias term. Similarly, for a conventional RNN-LM which usually takes words as inputs, if $w_t = k$, then the input to the RNN-LM at t is the *input embedding* $\mathbf{x}^k$, the k-th column of the embedding matrix $\mathbf{X} \in \mathbb{R}^{n \times |\mathcal{V}|}$. Our model simply replaces the input embeddings $\mathbf{X}$ with the output from a character-level convolutional neural network, to be described below.

If we denote $w_{1:T} = [w_1, \cdots, w_T]$ to be the sequence of words in the training corpus, training involves minimizing the negative log-likelihood (NLL) of the sequence

$$NLL = -\sum_{t=1}^{T} \log \Pr(w_t | w_{1:t-1}) \quad (4)$$

which is typically done by truncated backpropagation through time (Werbos 1990; Graves 2013).

Character-level Convolutional Neural Network

In our model, the input at time t is an output from a character-level convolutional neural network (CharCNN), which we describe in this section. CNNs (LeCun et al. 1989) have achieved state-of-the-art results on computer vision (Krizhevsky, Sutskever, and Hinton 2012) and have also been shown to be effective for various NLP tasks (Collobert et al. 2011). Architectures employed for NLP applications differ in that they typically involve temporal rather than spatial convolutions.

Let $\mathcal{C}$ be the vocabulary of characters, d be the dimensionality of character embeddings,[3] and $\mathbf{Q} \in \mathbb{R}^{d \times |\mathcal{C}|}$ be the matrix character embeddings. Suppose that word $k \in \mathcal{V}$ is made up of a sequence of characters $[c_1, \ldots, c_l]$, where l is the length of word k. Then the character-level representation of k is given by the matrix $\mathbf{C}^k \in \mathbb{R}^{d \times l}$, where the j-th column corresponds to the character embedding for c_j (i.e. the c_j-th column of $\mathbf{Q}$).[4]

[2] In our work, predictions are at the word-level, and hence we still utilize word embeddings in the output layer.

[3] Given that $|\mathcal{C}|$ is usually small, some authors work with one-hot representations of characters. However we found that using lower dimensional representations of characters (i.e. $d < |\mathcal{C}|$) performed slightly better.

[4] Two technical details warrant mention here: (1) we append start-of-word and end-of-word characters to each word to better represent prefixes and suffixes and hence $\mathbf{C}^k$ actually has $l + 2$ columns; (2) for batch processing, we zero-pad $\mathbf{C}^k$ so that the number of columns is constant (equal to the max word length) for all words in $\mathcal{V}$.

We apply a narrow convolution between $\mathbf{C}^k$ and a *filter* (or *kernel*) $\mathbf{H} \in \mathbb{R}^{d \times w}$ of width w, after which we add a bias and apply a nonlinearity to obtain a *feature map* $\mathbf{f}^k \in \mathbb{R}^{l-w+1}$. Specifically, the i-th element of $\mathbf{f}^k$ is given by:

$$\mathbf{f}^k[i] = \tanh(\langle \mathbf{C}^k[*, i : i + w - 1], \mathbf{H} \rangle + b) \quad (5)$$

where $\mathbf{C}^k[*, i : i+w-1]$ is the i-to-$(i+w-1)$-th column of $\mathbf{C}^k$ and $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{Tr}(\mathbf{A}\mathbf{B}^T)$ is the Frobenius inner product. Finally, we take the *max-over-time*

$$y^k = \max_i \mathbf{f}^k[i] \quad (6)$$

as the feature corresponding to the filter $\mathbf{H}$ (when applied to word k). The idea is to capture the most important feature—the one with the highest value—for a given filter. A filter is essentially picking out a character n-gram, where the size of the n-gram corresponds to the filter width.

We have described the process by which *one* feature is obtained from *one* filter matrix. Our CharCNN uses multiple filters of varying widths to obtain the feature vector for k. So if we have a total of h filters $\mathbf{H}_1, \ldots, \mathbf{H}_h$, then $\mathbf{y}^k = [y^k_1, \ldots, y^k_h]$ is the input representation of k. For many NLP applications h is typically chosen to be in $[100, 1000]$.

Highway Network

We could simply replace $\mathbf{x}^k$ (the word embedding) with $\mathbf{y}^k$ at each t in the RNN-LM, and as we show later, this simple model performs well on its own (Table 7). One could also have a multilayer perceptron (MLP) over $\mathbf{y}^k$ to model interactions between the character n-grams picked up by the filters, but we found that this resulted in worse performance.

Instead we obtained improvements by running $\mathbf{y}^k$ through a *highway network*, recently proposed by Srivastava et al. (2015). Whereas one layer of an MLP applies an affine transformation followed by a nonlinearity to obtain a new set of features,

$$\mathbf{z} = g(\mathbf{W}\mathbf{y} + \mathbf{b}) \quad (7)$$

one layer of a highway network does the following:

$$\mathbf{z} = \mathbf{t} \odot g(\mathbf{W}_H \mathbf{y} + \mathbf{b}_H) + (\mathbf{1} - \mathbf{t}) \odot \mathbf{y} \quad (8)$$

where g is a nonlinearity, $\mathbf{t} = \sigma(\mathbf{W}_T \mathbf{y} + \mathbf{b}_T)$ is called the *transform* gate, and $(\mathbf{1} - \mathbf{t})$ is called the *carry* gate. Similar to the memory cells in LSTM networks, highway layers allow for training of deep networks by adaptively *carrying* some dimensions of the input directly to the output.[5] By construction the dimensions of $\mathbf{y}$ and $\mathbf{z}$ have to match, and hence $\mathbf{W}_T$ and $\mathbf{W}_H$ are square matrices.

Experimental Setup

As is standard in language modeling, we use perplexity (PPL) to evaluate the performance of our models. Perplexity of a model over a sequence $[w_1, \ldots, w_T]$ is given by

$$PPL = \exp\Big(\frac{NLL}{T}\Big) \quad (9)$$

[5] Srivastava et al. (2015) recommend initializing $\mathbf{b}_T$ to a negative value, in order to militate the initial behavior towards *carry*. We initialized $\mathbf{b}_T$ to a small interval around -2.

	DATA-S			DATA-L		
	$\|\mathcal{V}\|$	$\|\mathcal{C}\|$	T	$\|\mathcal{V}\|$	$\|\mathcal{C}\|$	T
English (EN)	10 k	51	1 m	60 k	197	20 m
Czech (CS)	46 k	101	1 m	206 k	195	17 m
German (DE)	37 k	74	1 m	339 k	260	51 m
Spanish (ES)	27 k	72	1 m	152 k	222	56 m
French (FR)	25 k	76	1 m	137 k	225	57 m
Russian (RU)	62 k	62	1 m	497 k	111	25 m
Arabic (AR)	86 k	132	4 m	–	–	–

Table 1: Corpus statistics. $|\mathcal{V}|$ = word vocabulary size; $|\mathcal{C}|$ = character vocabulary size; T = number of tokens in training set. The small English data is from the Penn Treebank and the Arabic data is from the News-Commentary corpus. The rest are from the 2013 ACL Workshop on Machine Translation. $|\mathcal{C}|$ is large because of (rarely occurring) special characters.

		Small	Large
CNN	d	15	15
	w	$[1, 2, 3, 4, 5, 6]$	$[1, 2, 3, 4, 5, 6, 7]$
	h	$[25 \cdot w]$	$[\min\{200, 50 \cdot w\}]$
	f	tanh	tanh
Highway	l	1	2
	g	ReLU	ReLU
LSTM	l	2	2
	m	300	650

Table 2: Architecture of the small and large models. d = dimensionality of character embeddings; w = filter widths; h = number of filter matrices, as a function of filter width (so the large model has filters of width $[1, 2, 3, 4, 5, 6, 7]$ of size $[50, 100, 150, 200, 200, 200, 200]$ for a total of 1100 filters); f, g = nonlinearity functions; l = number of layers; m = number of hidden units.

where NLL is calculated over the test set. We test the model on corpora of varying languages and sizes (statistics available in Table 1).

We conduct hyperparameter search, model introspection, and ablation studies on the English Penn Treebank (PTB) (Marcus, Santorini, and Marcinkiewicz 1993), utilizing the standard training (0-20), validation (21-22), and test (23-24) splits along with pre-processing by Mikolov et al. (2010). With approximately 1m tokens and $|\mathcal{V}| = 10$k, this version has been extensively used by the language modeling community and is publicly available.[6]

With the optimal hyperparameters tuned on PTB, we apply the model to various morphologically rich languages: Czech, German, French, Spanish, Russian, and Arabic. Non-Arabic data comes from the 2013 ACL Workshop on Machine Translation,[7] and we use the same train/validation/test splits as in Botha and Blunsom (2014). While the raw data are publicly available, we obtained the preprocessed versions from the authors,[8] whose morphological NLM serves as a baseline for our work. We train on both the small datasets (DATA-S) with 1m tokens per language, and the large datasets (DATA-L) including the large English data which has a much bigger $|\mathcal{V}|$ than the PTB. Arabic data comes from the News-Commentary corpus,[9] and we perform our own preprocessing and train/validation/test splits.

In these datasets only singleton words were replaced with <unk> and hence we effectively use the full vocabulary. It is worth noting that the character model can utilize surface forms of OOV tokens (which were replaced with <unk>), but we do not do this and stick to the preprocessed versions (despite disadvantaging the character models) for exact comparison against prior work.

[6]http://www.fit.vutbr.cz/~imikolov/rnnlm/

[7]http://www.statmt.org/wmt13/translation-task.html

[8]http://bothameister.github.io/

[9]http://opus.lingfil.uu.se/News-Commentary.php

Optimization

The models are trained by truncated backpropagation through time (Werbos 1990; Graves 2013). We backpropagate for 35 time steps using stochastic gradient descent where the learning rate is initially set to 1.0 and halved if the perplexity does not decrease by more than 1.0 on the validation set after an epoch. On DATA-S we use a batch size of 20 and on DATA-L we use a batch size of 100 (for greater efficiency). Gradients are averaged over each batch. We train for 25 epochs on non-Arabic and 30 epochs on Arabic data (which was sufficient for convergence), picking the best performing model on the validation set. Parameters of the model are randomly initialized over a uniform distribution with support $[-0.05, 0.05]$.

For regularization we use dropout (Hinton et al. 2012) with probability 0.5 on the LSTM input-to-hidden layers (except on the initial Highway to LSTM layer) and the hidden-to-output softmax layer. We further constrain the norm of the gradients to be below 5, so that if the L_2 norm of the gradient exceeds 5 then we renormalize it to have $|| \cdot || = 5$ before updating. The gradient norm constraint was crucial in training the model. These choices were largely guided by previous work of Zaremba et al. (2014) on word-level language modeling with LSTMs.

Finally, in order to speed up training on DATA-L we employ a hierarchical softmax (Morin and Bengio 2005)—a common strategy for training language models with very large $|\mathcal{V}|$—instead of the usual softmax. We pick the number of clusters $c = \lceil \sqrt{|\mathcal{V}|} \rceil$ and randomly split $\mathcal{V}$ into mutually exclusive and collectively exhaustive subsets $\mathcal{V}_1, \ldots, \mathcal{V}_c$ of (approximately) equal size.[10] Then $\Pr(w_{t+1} = j | w_{1:t})$ be-

[10]While Brown clustering/frequency-based clustering is commonly used in the literature (e.g. Botha and Blunsom (2014) use Brown clusering), we used random clusters as our implementation enjoys the best speed-up when the number of words in each cluster is approximately equal. We found random clustering to work surprisingly well.

	PPL	Size
LSTM-Word-Small	97.6	5 m
LSTM-Char-Small	92.3	5 m
LSTM-Word-Large	85.4	20 m
LSTM-Char-Large	78.9	19 m
KN-5 (Mikolov et al. 2012)	141.2	2 m
RNN† (Mikolov et al. 2012)	124.7	6 m
RNN-LDA† (Mikolov et al. 2012)	113.7	7 m
genCNN† (Wang et al. 2015)	116.4	8 m
FOFE-FNNLM† (Zhang et al. 2015)	108.0	6 m
Deep RNN (Pascanu et al. 2013)	107.5	6 m
Sum-Prod Net† (Cheng et al. 2014)	100.0	5 m
LSTM-1† (Zaremba et al. 2014)	82.7	20 m
LSTM-2† (Zaremba et al. 2014)	78.4	52 m

Table 3: Performance of our model versus other neural language models on the English Penn Treebank test set. PPL refers to perplexity (lower is better) and size refers to the approximate number of parameters in the model. KN-5 is a Kneser-Ney 5-gram language model which serves as a non-neural baseline. †For these models the authors did not explicitly state the number of parameters, and hence sizes shown here are estimates based on our understanding of their papers or private correspondence with the respective authors.

comes,

$$\Pr(w_{t+1} = j | w_{1:t}) = \frac{\exp(\mathbf{h}_t \cdot \mathbf{s}^r + t^r)}{\sum_{r'=1}^{c} \exp(\mathbf{h}_t \cdot \mathbf{s}^{r'} + t^{r'})} \times \frac{\exp(\mathbf{h}_t \cdot \mathbf{p}_r^j + q_r^j)}{\sum_{j' \in \mathcal{V}_r} \exp(\mathbf{h}_t \cdot \mathbf{p}_r^{j'} + q_r^{j'})} \quad (10)$$

where r is the cluster index such that $j \in \mathcal{V}_r$. The first term is simply the probability of picking cluster r, and the second term is the probability of picking word j given that cluster r is picked. We found that hierarchical softmax was not necessary for models trained on DATA-S.

Results

English Penn Treebank

We train two versions of our model to assess the trade-off between performance and size. Architecture of the small (LSTM-Char-Small) and large (LSTM-Char-Large) models is summarized in Table 2. As another baseline, we also train two comparable LSTM models that use word embeddings only (LSTM-Word-Small, LSTM-Word-Large). LSTM-Word-Small uses 200 hidden units and LSTM-Word-Large uses 650 hidden units. Word embedding sizes are also 200 and 650 respectively. These were chosen to keep the number of parameters similar to the corresponding character-level model.

As can be seen from Table 3, our large model is on par with the existing state-of-the-art (Zaremba et al. 2014), despite having approximately 60% fewer parameters. Our small model significantly outperforms other NLMs of similar size, even though it is penalized by the fact that the dataset already has OOV words replaced with <unk> (other models are purely word-level models). While lower perplexities have been reported with model ensembles (Mikolov and Zweig 2012), we do not include them here as they are not comparable to the current work.

		DATA-S					
		CS	DE	ES	FR	RU	AR
Botha	KN-4	545	366	241	274	396	323
	MLBL	465	296	200	225	304	–
Small	Word	503	305	212	229	352	216
	Morph	414	278	197	216	290	230
	Char	401	260	182	189	278	196
Large	Word	493	286	200	222	357	172
	Morph	398	263	177	196	271	**148**
	Char	**371**	**239**	**165**	**184**	**261**	**148**

Table 4: Test set perplexities for DATA-S. First two rows are from Botha (2014) (except on Arabic where we trained our own KN-4 model) while the last six are from this paper. KN-4 is a Kneser-Ney 4-gram language model, and MLBL is the best performing morphological logbilinear model from Botha (2014). Small/Large refer to model size (see Table 2), and Word/Morph/Char are models with words/morphemes/characters as inputs respectively.

Other Languages

The model's performance on the English PTB is informative to the extent that it facilitates comparison against the large body of existing work. However, English is relatively simple from a morphological standpoint, and thus our next set of results (and arguably the main contribution of this paper) is focused on languages with richer morphology (Table 4, Table 5).

We compare our results against the morphological logbilinear (MLBL) model from Botha and Blunsom (2014), whose model also takes into account subword information through morpheme embeddings that are summed at the input and output layers. As comparison against the MLBL models is confounded by our use of LSTMs—widely known to outperform their feed-forward/log-bilinear cousins—we also train an LSTM version of the morphological NLM, where the input representation of a word given to the LSTM is a summation of the word's morpheme embeddings. Concretely, suppose that $\mathcal{M}$ is the set of morphemes in a language, $\mathbf{M} \in \mathbb{R}^{n \times |\mathcal{M}|}$ is the matrix of morpheme embeddings, and $\mathbf{m}^j$ is the j-th column of $\mathbf{M}$ (i.e. a morpheme embedding). Given the input word k, we feed the following representation to the LSTM:

$$\mathbf{x}^k + \sum_{j \in \mathcal{M}_k} \mathbf{m}^j \quad (11)$$

where $\mathbf{x}^k$ is the word embedding (as in a word-level model) and $\mathcal{M}_k \subset \mathcal{M}$ is the set of morphemes for word k. The

		DATA-L					
		CS	DE	ES	FR	RU	EN
Botha	KN-4	862	463	219	243	390	291
	MLBL	643	404	203	227	**300**	273
Small	Word	701	347	186	202	353	236
	Morph	615	331	189	209	331	233
	Char	**578**	**305**	**169**	**190**	313	**216**

Table 5: Test set perplexities on DATA-L. First two rows are from Botha (2014), while the last three rows are from the small LSTM models described in the paper. KN-4 is a Kneser-Ney 4-gram language model, and MLBL is the best performing morphological logbilinear model from Botha (2014). Word/Morph/Char are models with words/morphemes/characters as inputs respectively.

morphemes are obtained by running an unsupervised morphological tagger as a preprocessing step.[11] We emphasize that the word embedding itself (i.e. $\mathbf{x}^k$) is added on top of the morpheme embeddings, as was done in Botha and Blunsom (2014). The morpheme embeddings are of size 200/650 for the small/large models respectively. We further train word-level LSTM models as another baseline.

On DATA-S it is clear from Table 4 that the character-level models outperform their word-level counterparts despite, again, being smaller.[12] The character models also outperform their morphological counterparts (both MLBL and LSTM architectures), although improvements over the morphological LSTMs are more measured. Note that the morpheme models have strictly more parameters than the word models because word embeddings are used as part of the input.

Due to memory constraints[13] we only train the small models on DATA-L (Table 5). Interestingly we do not observe significant differences going from word to morpheme LSTMs on Spanish, French, and English. The character models again outperform the word/morpheme models. We also observe significant perplexity reductions even on English when $\mathcal{V}$ is large. We conclude this section by noting that we used the same architecture for all languages and did not perform any language-specific tuning of hyperparameters.

Discussion

Learned Word Representations

We explore the word representations learned by the models on the PTB. Table 6 has the nearest neighbors of word representations learned from both the word-level and character-level models. For the character models we compare the representations obtained before and after highway layers.

Before the highway layers the representations seem to solely rely on surface forms—for example the nearest neighbors of *you* are *your, young, four, youth*, which are close to *you* in terms of edit distance. The highway layers however, seem to enable encoding of semantic features that are not discernable from orthography alone. After highway layers the nearest neighbor of *you* is *we*, which is orthographically distinct from *you*. Another example is *while* and *though*—these words are far apart edit distance-wise yet the composition model is able to place them near each other. The model also makes some clear mistakes (e.g. *his* and *hhs*), highlighting the limits of our approach, although this could be due to the small dataset.

The learned representations of OOV words (*computer-aided*, *misinformed*) are positioned near words with the same part-of-speech. The model is also able to correct for incorrect/non-standard spelling (*looooook*), indicating potential applications for text normalization in noisy domains.

Learned Character N-gram Representations

As discussed previously, each filter of the CharCNN is essentially learning to detect particular character n-grams. Our initial expectation was that each filter would learn to activate on different morphemes and then build up semantic representations of words from the identified morphemes. However, upon reviewing the character n-grams picked up by the filters (i.e. those that maximized the value of the filter), we found that they did not (in general) correspond to valid morphemes.

To get a better intuition for what the character composition model is learning, we plot the learned representations of all character n-grams (that occurred as part of at least two words in $\mathcal{V}$) via principal components analysis (Figure 2). We feed each character n-gram into the CharCNN and use the CharCNN's output as the fixed dimensional representation for the corresponding character n-gram. As is apparent from Figure 2, the model learns to differentiate between prefixes (red), suffixes (blue), and others (grey). We also find that the representations are particularly sensitive to character n-grams containing hyphens (orange), presumably because this is a strong signal of a word's part-of-speech.

Highway Layers

We quantitatively investigate the effect of highway network layers via ablation studies (Table 7). We train a model without any highway layers, and find that performance decreases significantly. As the difference in performance could be due to the decrease in model size, we also train a model that feeds $\mathbf{y}^k$ (i.e. word representation from the CharCNN) through a one-layer multilayer perceptron (MLP) to use as input into the LSTM. We find that the MLP does poorly, although this could be due to optimization issues.

We hypothesize that highway networks are especially well-suited to work with CNNs, adaptively combining local features detected by the individual filters. CNNs have already proven to be been successful for many NLP tasks

[11] We use *Morfessor Cat-MAP* (Creutz and Lagus 2007), as in Botha and Blunsom (2014).

[12] The difference in parameters is greater for non-PTB corpora as the size of the word model scales faster with $|\mathcal{V}|$. For example, on Arabic the small/large word models have 35m/121m parameters while the corresponding character models have 29m/69m parameters respectively.

[13] All models were trained on GPUs with 2GB memory.

	In Vocabulary					Out-of-Vocabulary		
	while	*his*	*you*	*richard*	*trading*	*computer-aided*	*misinformed*	*looooook*
LSTM-Word	*although*	*your*	*conservatives*	*jonathan*	*advertised*	–	–	–
	letting	*her*	*we*	*robert*	*advertising*	–	–	–
	though	*my*	*guys*	*neil*	*turnover*	–	–	–
	minute	*their*	*i*	*nancy*	*turnover*	–	–	–
LSTM-Char (before highway)	*chile*	*this*	*your*	*hard*	*heading*	*computer-guided*	*informed*	*look*
	whole	*hhs*	*young*	*rich*	*training*	*computerized*	*performed*	*cook*
	meanwhile	*is*	*four*	*richer*	*reading*	*disk-drive*	*transformed*	*looks*
	white	*has*	*youth*	*richter*	*leading*	*computer*	*inform*	*shook*
LSTM-Char (after highway)	*meanwhile*	*hhs*	*we*	*eduard*	*trade*	*computer-guided*	*informed*	*look*
	whole	*this*	*your*	*gerard*	*training*	*computer-driven*	*performed*	*looks*
	though	*their*	*doug*	*edward*	*traded*	*computerized*	*outperformed*	*looked*
	nevertheless	*your*	*i*	*carl*	*trader*	*computer*	*transformed*	*looking*

Table 6: Nearest neighbor words (based on cosine similarity) of word representations from the large word-level and character-level (before and after highway layers) models trained on the PTB. Last three words are OOV words, and therefore they do not have representations in the word-level model.

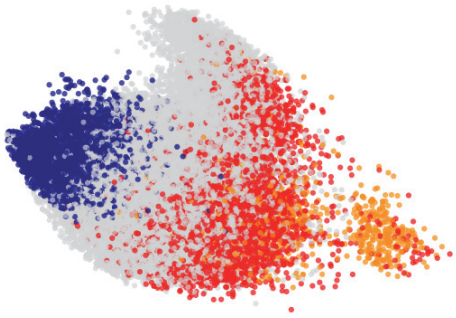


Figure 2: Plot of character n-gram representations via PCA for English. Colors correspond to: prefixes (red), suffixes (blue), hyphenated (orange), and all others (grey). Prefixes refer to character n-grams which start with the start-of-word character. Suffixes likewise refer to character n-grams which end with the end-of-word character.

(Collobert et al. 2011; Shen et al. 2014; Kalchbrenner, Grefenstette, and Blunsom 2014; Kim 2014; Zhang, Zhao, and LeCun 2015; Lei, Barzilay, and Jaakola 2015), and we posit that further gains could be achieved by employing highway layers on top of existing CNN architectures.

We also anecdotally note that (1) having one to two highway layers was important, but more highway layers generally resulted in similar performance (though this may depend on the size of the datasets), (2) having more convolutional layers before max-pooling did not help, and (3) highway layers did not improve models that only used word embeddings as inputs.

Effect of Corpus/Vocab Sizes

We next study the effect of training corpus/vocabulary sizes on the relative performance between the different models. We take the German (DE) dataset from DATA-L and vary the training corpus/vocabulary sizes, calculating the perplexity reductions as a result of going from a small word-level model to a small character-level model. To vary the vocabulary size we take the most frequent k words and replace the rest with <unk>. As with previous experiments the character model does not utilize surface forms of <unk> and simply treats it as another token. Although Table 8 suggests that the perplexity reductions become less pronounced as the corpus size increases, we nonetheless find that the character-level model outperforms the word-level model in all scenarios.

	LSTM-Char	
	Small	Large
No Highway Layers	100.3	84.6
One Highway Layer	92.3	79.7
Two Highway Layers	90.1	78.9
One MLP Layer	111.2	92.6

Table 7: Perplexity on the Penn Treebank for small/large models trained with/without highway layers.

		$\lvert\mathcal{V}\rvert$			
		10 k	25 k	50 k	100 k
T	1 m	17%	16%	21%	–
	5 m	8%	14%	16%	21%
	10 m	9%	9%	12%	15%
	25 m	9%	8%	9%	10%

Table 8: Perplexity reductions by going from small word-level to character-level models based on different corpus/vocabulary sizes on German (DE). $|\mathcal{V}|$ is the vocabulary size and T is the number of tokens in the training set. The full vocabulary of the 1m dataset was less than 100k and hence that scenario is unavailable.

Further Observations

We report on some further experiments and observations:

- Combining word embeddings with the CharCNN's output to form a combined representation of a word (to be used as input to the LSTM) resulted in slightly worse performance (81 on PTB with a large model). This was surprising, as improvements have been reported on part-of-speech tagging (dos Santos and Zadrozny 2014) and named entity recognition (dos Santos and Guimaraes 2015) by concatenating word embeddings with the output from a character-level CNN. While this could be due to insufficient experimentation on our part,[14] it suggests that for some tasks, word embeddings are superfluous—character inputs are good enough.
- While our model requires additional convolution operations over characters and is thus slower than a comparable word-level model which can perform a simple lookup at the input layer, we found that the difference was manageable with optimized GPU implementations—for example on PTB the large character-level model trained at 1500 tokens/sec compared to the word-level model which trained at 3000 tokens/sec. For scoring, our model can have the same running time as a pure word-level model, as the CharCNN's outputs can be pre-computed for all words in $\mathcal{V}$. This would, however, be at the expense of increased model size, and thus a trade-off can be made between run-time speed and memory (e.g. one could restrict the pre-computation to the most frequent words).

Related Work

Neural Language Models (NLM) encompass a rich family of neural network architectures for language modeling. Some example architectures include feed-forward (Bengio, Ducharme, and Vincent 2003), recurrent (Mikolov et al. 2010), sum-product (Cheng et al. 2014), log-bilinear (Mnih and Hinton 2007), and convolutional (Wang et al. 2015) networks.

In order to address the rare word problem, Alexandrescu and Kirchhoff (2006)—building on analogous work on count-based n-gram language models by Bilmes and Kirchhoff (2003)—represent a word as a set of shared factor embeddings. Their Factored Neural Language Model (FNLM) can incorporate morphemes, word shape information (e.g. capitalization) or any other annotation (e.g. part-of-speech tags) to represent words.

A specific class of FNLMs leverages morphemic information by viewing a word as a function of its (learned) morpheme embeddings (Luong, Socher, and Manning 2013; Botha and Blunsom 2014; Qui et al. 2014). For example Luong, Socher, and Manning (2013) apply a recursive neural network over morpheme embeddings to obtain the embedding for a single word. While such models have proved useful, they require morphological tagging as a preprocessing step.

[14] We experimented with (1) concatenation, (2) tensor products, (3) averaging, and (4) adaptive weighting schemes whereby the model learns a convex combination of word embeddings and the CharCNN outputs.

Another direction of work has involved purely character-level NLMs, wherein both input and output are characters (Sutskever, Martens, and Hinton 2011; Graves 2013). Character-level models obviate the need for morphological tagging or manual feature engineering, and have the attractive property of being able to generate novel words. However they are generally outperformed by word-level models (Mikolov et al. 2012).

Outside of language modeling, improvements have been reported on part-of-speech tagging (dos Santos and Zadrozny 2014) and named entity recognition (dos Santos and Guimaraes 2015) by representing a word as a concatenation of its word embedding and an output from a character-level CNN, and using the combined representation as features in a Conditional Random Field (CRF). Zhang, Zhao, and LeCun (2015) do away with word embeddings completely and show that for text classification, a deep CNN over characters performs well. Ballesteros, Dyer, and Smith (2015) use an RNN over characters only to train a transition-based parser, obtaining improvements on many morphologically rich languages.

Finally, Ling et al. (2015) apply a bi-directional LSTM over characters to use as inputs for language modeling and part-of-speech tagging. They show improvements on various languages (English, Portuguese, Catalan, German, Turkish). It remains open as to which character composition model (i.e. CNN or LSTM) performs better.

Conclusion

We have introduced a neural language model that utilizes only character-level inputs. Predictions are still made at the word-level. Despite having fewer parameters, our model outperforms baseline models that utilize word/morpheme embeddings in the input layer. Our work questions the necessity of word embeddings (as inputs) for neural language modeling.

Analysis of word representations obtained from the character composition part of the model further indicates that the model is able to encode, from characters only, rich semantic and orthographic features. Using the CharCNN and highway layers for representation learning (e.g. as input into word2vec (Mikolov et al. 2013)) remains an avenue for future work.

Insofar as sequential processing of words as inputs is ubiquitous in natural language processing, it would be interesting to see if the architecture introduced in this paper is viable for other tasks—for example, as an encoder/decoder in neural machine translation (Cho et al. 2014; Sutskever, Vinyals, and Le 2014).

Acknowledgments

We are especially grateful to Jan Botha for providing the preprocessed datasets and the model results.

References

Alexandrescu, A., and Kirchhoff, K. 2006. Factored Neural Language Models. In *Proceedings of NAACL.*

Ballesteros, M.; Dyer, C.; and Smith, N. A. 2015. Improved Transition-Based Parsing by Modeling Characters instead of Words with LSTMs. In *Proceedings of EMNLP*.

Bengio, Y.; Ducharme, R.; and Vincent, P. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research* 3:1137–1155.

Bengio, Y.; Simard, P.; and Frasconi, P. 1994. Learning Long-term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks* 5:157–166.

Bilmes, J., and Kirchhoff, K. 2003. Factored Language Models and Generalized Parallel Backoff. In *Proceedings of NAACL*.

Botha, J., and Blunsom, P. 2014. Compositional Morphology for Word Representations and Language Modelling. In *Proceedings of ICML*.

Botha, J. 2014. Probabilistic Modelling of Morphologically Rich Languages. *DPhil Dissertation, Oxford University*.

Chen, S., and Goodman, J. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. *Technical Report, Harvard University*.

Cheng, W. C.; Kok, S.; Pham, H. V.; Chieu, H. L.; and Chai, K. M. 2014. Language Modeling with Sum-Product Networks. In *Proceedings of INTERSPEECH*.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of EMNLP*.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research* 12:2493–2537.

Creutz, M., and Lagus, K. 2007. Unsupervised Models for Morpheme Segmentation and Morphology Learning. In *Proceedings of the ACM Transations on Speech and Language Processing*.

Deerwester, S.; Dumais, S.; and Harshman, R. 1990. Indexing by Latent Semantic Analysis. *Journal of American Society of Information Science* 41:391–407.

dos Santos, C. N., and Guimaraes, V. 2015. Boosting Named Entity Recognition with Neural Character Embeddings. In *Proceedings of ACL Named Entities Workshop*.

dos Santos, C. N., and Zadrozny, B. 2014. Learning Character-level Representations for Part-of-Speech Tagging. In *Proceedings of ICML*.

Graves, A. 2013. Generating Sequences with Recurrent Neural Networks. *arXiv:1308.0850*.

Hinton, G.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2012. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. *arxiv:1207.0580*.

Hochreiter, S., and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9:1735–1780.

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of ACL*.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of EMNLP*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of NIPS*.

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; and Jackel, L. D. 1989. Handwritten Digit Recognition with a Backpropagation Network. In *Proceedings of NIPS*.

Lei, T.; Barzilay, R.; and Jaakola, T. 2015. Molding CNNs for Text: Non-linear, Non-consecutive Convolutions. In *Proceedings of EMNLP*.

Ling, W.; Lui, T.; Marujo, L.; Astudillo, R. F.; Amir, S.; Dyer, C.; Black, A. W.; and Trancoso, I. 2015. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In *Proceedings of EMNLP*.

Luong, M.-T.; Socher, R.; and Manning, C. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of CoNLL*.

Marcus, M.; Santorini, B.; and Marcinkiewicz, M. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics* 19:331–330.

Mikolov, T., and Zweig, G. 2012. Context Dependent Recurrent Neural Network Language Model. In *Proceedings of SLT*.

Mikolov, T.; Karafiat, M.; Burget, L.; Cernocky, J.; and Khudanpur, S. 2010. Recurrent Neural Network Based Language Model. In *Proceedings of INTERSPEECH*.

Mikolov, T.; Deoras, A.; Kombrink, S.; Burget, L.; and Cernocky, J. 2011. Empirical Evaluation and Combination of Advanced Language Modeling Techniques. In *Proceedings of INTERSPEECH*.

Mikolov, T.; Sutskever, I.; Deoras, A.; Le, H.-S.; Kombrink, S.; and Cernocky, J. 2012. Subword Language Modeling with Neural Networks. *preprint: www.fit.vutbr.cz/imikolov/rnnlm/char.pdf*.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781*.

Mnih, A., and Hinton, G. 2007. Three New Graphical Models for Statistical Language Modelling. In *Proceedings of ICML*.

Morin, F., and Bengio, Y. 2005. Hierarchical Probabilistic Neural Network Language Model. In *Proceedings of AISTATS*.

Pascanu, R.; Culcehre, C.; Cho, K.; and Bengio, Y. 2013. How to Construct Deep Neural Networks. *arXiv:1312.6026*.

Qui, S.; Cui, Q.; Bian, J.; and Gao, B. 2014. Co-learning of Word Representations and Morpheme Representations. In *Proceedings of COLING*.

Shen, Y.; He, X.; Gao, J.; Deng, L.; and Mesnil, G. 2014. A Latent Semantic Model with Convolutional-pooling Structure for Information Retrieval. In *Proceedings of CIKM*.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Training Very Deep Networks. *arXiv:1507.06228*.

Sundermeyer, M.; Schluter, R.; and Ney, H. 2012. LSTM Neural Networks for Language Modeling.

Sutskever, I.; Martens, J.; and Hinton, G. 2011. Generating Text with Recurrent Neural Networks.

Sutskever, I.; Vinyals, O.; and Le, Q. 2014. Sequence to Sequence Learning with Neural Networks.

Wang, M.; Lu, Z.; Li, H.; Jiang, W.; and Liu, Q. 2015. *gen*CNN: A Convolutional Architecture for Word Sequence Prediction. In *Proceedings of ACL*.

Werbos, P. 1990. Back-propagation Through Time: what it does and how to do it. In *Proceedings of IEEE*.

Zaremba, W.; Sutskever, I.; and Vinyals, O. 2014. Recurrent Neural Network Regularization. *arXiv:1409.2329*.

Zhang, S.; Jiang, H.; Xu, M.; Hou, J.; and Dai, L. 2015. The Fixed-Size Ordinally-Forgetting Encoding Method for Neural Network Language Models. In *Proceedings of ACL*.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level Convolutional Networks for Text Classification. In *Proceedings of NIPS*.

End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF

Xuezhe Ma and **Eduard Hovy**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
xuezhem@cs.cmu.edu, ehovy@cmu.edu

Abstract

State-of-the-art sequence labeling systems traditionally require large amounts of task-specific knowledge in the form of hand-crafted features and data pre-processing. In this paper, we introduce a novel neutral network architecture that benefits from both word- and character-level representations automatically, by using combination of bidirectional LSTM, CNN and CRF. Our system is truly end-to-end, requiring no feature engineering or data pre-processing, thus making it applicable to a wide range of sequence labeling tasks. We evaluate our system on two data sets for two sequence labeling tasks — Penn Treebank WSJ corpus for part-of-speech (POS) tagging and CoNLL 2003 corpus for named entity recognition (NER). We obtain state-of-the-art performance on both datasets — 97.55% accuracy for POS tagging and 91.21% F1 for NER.

1 Introduction

Linguistic sequence labeling, such as part-of-speech (POS) tagging and named entity recognition (NER), is one of the first stages in deep language understanding and its importance has been well recognized in the natural language processing community. Natural language processing (NLP) systems, like syntactic parsing (Nivre and Scholz, 2004; McDonald et al., 2005; Koo and Collins, 2010; Ma and Zhao, 2012a; Ma and Zhao, 2012b; Chen and Manning, 2014; Ma and Hovy, 2015) and entity coreference resolution (Ng, 2010; Ma et al., 2016), are becoming more sophisticated, in part because of utilizing output information of POS tagging or NER systems.

Most traditional high performance sequence labeling models are linear statistical models, including Hidden Markov Models (HMM) and Conditional Random Fields (CRF) (Ratinov and Roth, 2009; Passos et al., 2014; Luo et al., 2015), which rely heavily on hand-crafted features and task-specific resources. For example, English POS taggers benefit from carefully designed word spelling features; orthographic features and external resources such as gazetteers are widely used in NER. However, such task-specific knowledge is costly to develop (Ma and Xia, 2014), making sequence labeling models difficult to adapt to new tasks or new domains.

In the past few years, non-linear neural networks with as input distributed word representations, also known as word embeddings, have been broadly applied to NLP problems with great success. Collobert et al. (2011) proposed a simple but effective feed-forward neutral network that independently classifies labels for each word by using contexts within a window with fixed size. Recently, recurrent neural networks (RNN) (Goller and Kuchler, 1996), together with its variants such as long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) and gated recurrent unit (GRU) (Cho et al., 2014), have shown great success in modeling sequential data. Several RNN-based neural network models have been proposed to solve sequence labeling tasks like speech recognition (Graves et al., 2013), POS tagging (Huang et al., 2015) and NER (Chiu and Nichols, 2015; Hu et al., 2016), achieving competitive performance against traditional models. However, even systems that have utilized distributed representations as inputs have used these to augment, rather than replace, hand-crafted features (e.g. word spelling and capitalization patterns). Their performance drops rapidly when the models solely depend on neural embeddings.

In this paper, we propose a neural network architecture for sequence labeling. It is a truly end-to-end model requiring no task-specific resources, feature engineering, or data pre-processing beyond pre-trained word embeddings on unlabeled corpora. Thus, our model can be easily applied to a wide range of sequence labeling tasks on different languages and domains. We first use convolutional neural networks (CNNs) (LeCun et al., 1989) to encode character-level information of a word into its character-level representation. Then we combine character- and word-level representations and feed them into bi-directional LSTM (BLSTM) to model context information of each word. On top of BLSTM, we use a sequential CRF to jointly decode labels for the whole sentence. We evaluate our model on two linguistic sequence labeling tasks — POS tagging on Penn Treebank WSJ (Marcus et al., 1993), and NER on English data from the CoNLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003). Our end-to-end model outperforms previous state-of-the-art systems, obtaining 97.55% accuracy for POS tagging and 91.21% F1 for NER. The contributions of this work are (i) proposing a novel neural network architecture for linguistic sequence labeling. (ii) giving empirical evaluations of this model on benchmark data sets for two classic NLP tasks. (iii) achieving state-of-the-art performance with this truly end-to-end system.

2 Neural Network Architecture

In this section, we describe the components (layers) of our neural network architecture. We introduce the neural layers in our neural network one-by-one from bottom to top.

2.1 CNN for Character-level Representation

Previous studies (Santos and Zadrozny, 2014; Chiu and Nichols, 2015) have shown that CNN is an effective approach to extract morphological information (like the prefix or suffix of a word) from characters of words and encode it into neural representations. Figure 1 shows the CNN we use to extract character-level representation of a given word. The CNN is similar to the one in Chiu and Nichols (2015), except that we use only character embeddings as the inputs to CNN, without character type features. A dropout layer (Srivastava et al., 2014) is applied before character embeddings are input to CNN.

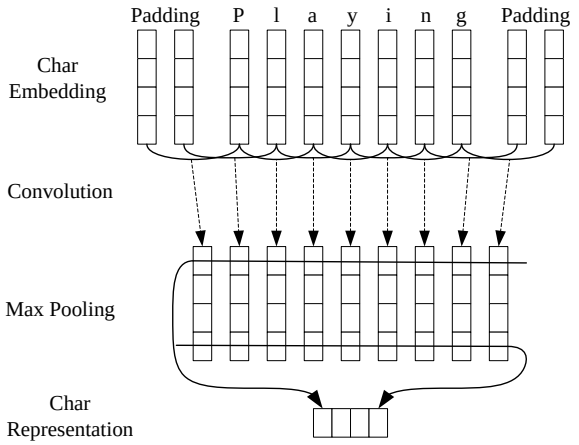


Figure 1: The convolution neural network for extracting character-level representations of words. Dashed arrows indicate a dropout layer applied before character embeddings are input to CNN.

2.2 Bi-directional LSTM

2.2.1 LSTM Unit

Recurrent neural networks (RNNs) are a powerful family of connectionist models that capture time dynamics via cycles in the graph. Though, in theory, RNNs are capable to capturing long-distance dependencies, in practice, they fail due to the gradient vanishing/exploding problems (Bengio et al., 1994; Pascanu et al., 2012).

LSTMs (Hochreiter and Schmidhuber, 1997) are variants of RNNs designed to cope with these gradient vanishing problems. Basically, a LSTM unit is composed of three multiplicative gates which control the proportions of information to forget and to pass on to the next time step. Figure 2 gives the basic structure of an LSTM unit.

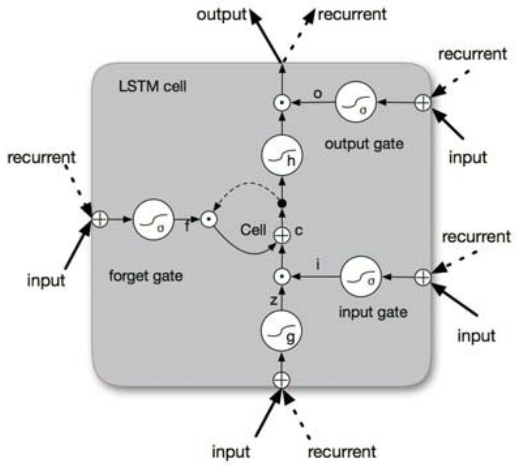


Figure 2: Schematic of LSTM unit.

Formally, the formulas to update an LSTM unit at time t are:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\boldsymbol{W}_i\mathbf{h}_{t-1} + \boldsymbol{U}_i\mathbf{x}_t + \boldsymbol{b}_i) \\
\mathbf{f}_t &= \sigma(\boldsymbol{W}_f\mathbf{h}_{t-1} + \boldsymbol{U}_f\mathbf{x}_t + \boldsymbol{b}_f) \\
\tilde{\mathbf{c}}_t &= \tanh(\boldsymbol{W}_c\mathbf{h}_{t-1} + \boldsymbol{U}_c\mathbf{x}_t + \boldsymbol{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{o}_t &= \sigma(\boldsymbol{W}_o\mathbf{h}_{t-1} + \boldsymbol{U}_o\mathbf{x}_t + \boldsymbol{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

where σ is the element-wise sigmoid function and $\odot$ is the element-wise product. $\mathbf{x}_t$ is the input vector (e.g. word embedding) at time t, and $\mathbf{h}_t$ is the hidden state (also called output) vector storing all the useful information at (and before) time t. $\boldsymbol{U}_i, \boldsymbol{U}_f, \boldsymbol{U}_c, \boldsymbol{U}_o$ denote the weight matrices of different gates for input $\mathbf{x}_t$, and $\boldsymbol{W}_i, \boldsymbol{W}_f, \boldsymbol{W}_c, \boldsymbol{W}_o$ are the weight matrices for hidden state $\mathbf{h}_t$. $\boldsymbol{b}_i, \boldsymbol{b}_f, \boldsymbol{b}_c, \boldsymbol{b}_o$ denote the bias vectors. It should be noted that we do not include peephole connections (Gers et al., 2003) in the our LSTM formulation.

2.2.2 BLSTM

For many sequence labeling tasks it is beneficial to have access to both past (left) and future (right) contexts. However, the LSTM's hidden state $\mathbf{h}_t$ takes information only from past, knowing nothing about the future. An elegant solution whose effectiveness has been proven by previous work (Dyer et al., 2015) is bi-directional LSTM (BLSTM). The basic idea is to present each sequence forwards and backwards to two separate hidden states to capture past and future information, respectively. Then the two hidden states are concatenated to form the final output.

2.3 CRF

For sequence labeling (or general structured prediction) tasks, it is beneficial to consider the correlations between labels in neighborhoods and jointly decode the best chain of labels for a given input sentence. For example, in POS tagging an adjective is more likely to be followed by a noun than a verb, and in NER with standard BIO2 annotation (Tjong Kim Sang and Veenstra, 1999) I-ORG cannot follow I-PER. Therefore, we model label sequence jointly using a conditional random field (CRF) (Lafferty et al., 2001), instead of decoding each label independently.

Formally, we use $\mathbf{z} = \{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$ to represent a generic input sequence where $\mathbf{z}_i$ is the input vector of the ith word. $\boldsymbol{y} = \{y_1, \cdots, y_n\}$ represents a generic sequence of labels for $\mathbf{z}$. $\mathcal{Y}(\mathbf{z})$ denotes the set of possible label sequences for $\mathbf{z}$. The probabilistic model for sequence CRF defines a family of conditional probability $p(\boldsymbol{y}|\mathbf{z}; \mathbf{W}, \mathbf{b})$ over all possible label sequences $\boldsymbol{y}$ given $\mathbf{z}$ with the following form:

$$p(\boldsymbol{y}|\mathbf{z}; \mathbf{W}, \mathbf{b}) = \frac{\prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, \mathbf{z})}{\sum_{y' \in \mathcal{Y}(\mathbf{z})} \prod_{i=1}^{n} \psi_i(y'_{i-1}, y'_i, \mathbf{z})}$$

where $\psi_i(y', y, \mathbf{z}) = \exp(\mathbf{W}_{y',y}^T \mathbf{z}_i + \mathbf{b}_{y',y})$ are potential functions, and $\mathbf{W}_{y',y}^T$ and $\mathbf{b}_{y',y}$ are the weight vector and bias corresponding to label pair (y', y), respectively.

For CRF training, we use the maximum conditional likelihood estimation. For a training set $\{(\mathbf{z}_i, \boldsymbol{y}_i)\}$, the logarithm of the likelihood (a.k.a. the log-likelihood) is given by:

$$L(\mathbf{W}, \mathbf{b}) = \sum_i \log p(\boldsymbol{y}|\mathbf{z}; \mathbf{W}, \mathbf{b})$$

Maximum likelihood training chooses parameters such that the log-likelihood $L(\mathbf{W}, \mathbf{b})$ is maximized.

Decoding is to search for the label sequence $\boldsymbol{y}^*$ with the highest conditional probability:

$$\boldsymbol{y}^* = \operatorname*{argmax}_{y \in \mathcal{Y}(\mathbf{z})} p(\boldsymbol{y}|\mathbf{z}; \mathbf{W}, \mathbf{b})$$

For a sequence CRF model (only interactions between two successive labels are considered), training and decoding can be solved efficiently by adopting the Viterbi algorithm.

2.4 BLSTM-CNNs-CRF

Finally, we construct our neural network model by feeding the output vectors of BLSTM into a CRF layer. Figure 3 illustrates the architecture of our network in detail.

For each word, the character-level representation is computed by the CNN in Figure 1 with character embeddings as inputs. Then the character-level representation vector is concatenated with the word embedding vector to feed into the BLSTM network. Finally, the output vectors of BLSTM are fed to the CRF layer to jointly decode the best label sequence. As shown in Figure 3, dropout layers are applied on both the input and output vectors of BLSTM. Experimental results show that using dropout significantly

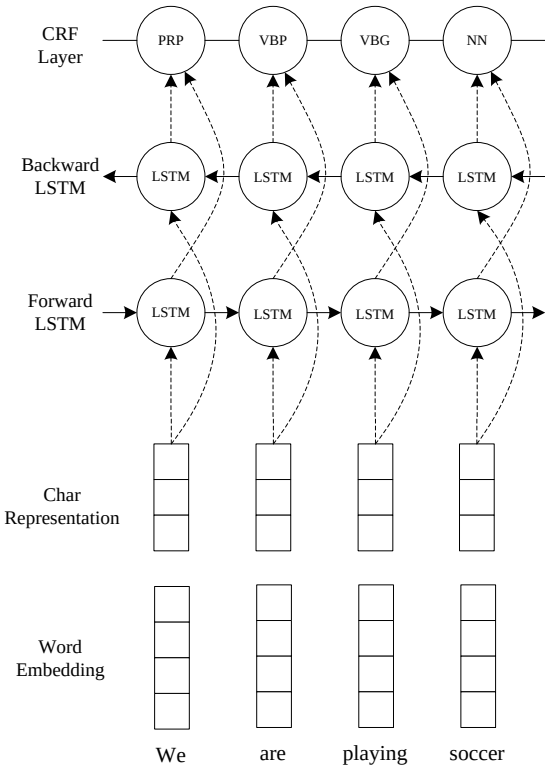


Figure 3: The main architecture of our neural network. The character representation for each word is computed by the CNN in Figure 1. Then the character representation vector is concatenated with the word embedding before feeding into the BLSTM network. Dashed arrows indicate dropout layers applied on both the input and output vectors of BLSTM.

improve the performance of our model (see Section 4.5 for details).

3 Network Training

In this section, we provide details about training the neural network. We implement the neural network using the Theano library (Bergstra et al., 2010). The computations for a single model are run on a GeForce GTX TITAN X GPU. Using the settings discussed in this section, the model training requires about 12 hours for POS tagging and 8 hours for NER.

3.1 Parameter Initialization

Word Embeddings. We use Stanford's publicly available GloVe 100-dimensional embeddings[1] trained on 6 billion words from Wikipedia and web text (Pennington et al., 2014)

We also run experiments on two other sets of published embeddings, namely Senna 50-dimensional embeddings[2] trained on Wikipedia and Reuters RCV-1 corpus (Collobert et al., 2011), and Google's Word2Vec 300-dimensional embeddings[3] trained on 100 billion words from Google News (Mikolov et al., 2013). To test the effectiveness of pretrained word embeddings, we experimented with randomly initialized embeddings with 100 dimensions, where embeddings are uniformly sampled from range $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}]$ where dim is the dimension of embeddings (He et al., 2015). The performance of different word embeddings is discussed in Section 4.4.

Character Embeddings. Character embeddings are initialized with uniform samples from $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}]$, where we set $dim = 30$.

Weight Matrices and Bias Vectors. Matrix parameters are randomly initialized with uniform samples from $[-\sqrt{\frac{6}{r+c}}, +\sqrt{\frac{6}{r+c}}]$, where r and c are the number of of rows and columns in the structure (Glorot and Bengio, 2010). Bias vectors are initialized to zero, except the bias $\mathbf{b}_f$ for the forget gate in LSTM , which is initialized to 1.0 (Jozefowicz et al., 2015).

3.2 Optimization Algorithm

Parameter optimization is performed with mini-batch stochastic gradient descent (SGD) with batch size 10 and momentum 0.9. We choose an initial learning rate of η_0 ($\eta_0 = 0.01$ for POS tagging, and 0.015 for NER, see Section 3.3), and the learning rate is updated on each epoch of training as $\eta_t = \eta_0/(1 + \rho t)$, with decay rate $\rho = 0.05$ and t is the number of epoch completed. To reduce the effects of "gradient exploding", we use a gradient clipping of 5.0 (Pascanu et al., 2012). We explored other more sophisticated optimization algorithms such as AdaDelta (Zeiler, 2012), Adam (Kingma and Ba, 2014) or RMSProp (Dauphin et al., 2015), but none of them meaningfully improve upon SGD with momentum and gradient clipping in our preliminary experiments.

Early Stopping. We use early stopping (Giles, 2001; Graves et al., 2013) based on performance on validation sets. The "best" parameters appear at around 50 epochs, according to our experiments.

[1] http://nlp.stanford.edu/projects/glove/

[2] http://ronan.collobert.com/senna/

[3] https://code.google.com/archive/p/word2vec/

Layer	Hyper-parameter	POS	NER
CNN	window size	3	3
	number of filters	30	30
LSTM	state size	200	200
	initial state	0.0	0.0
	peepholes	no	no
Dropout	dropout rate	0.5	0.5
	batch size	10	10
	initial learning rate	0.01	0.015
	decay rate	0.05	0.05
	gradient clipping	5.0	5.0

Table 1: Hyper-parameters for all experiments.

Fine Tuning. For each of the embeddings, we fine-tune initial embeddings, modifying them during gradient updates of the neural network model by back-propagating gradients. The effectiveness of this method has been previously explored in sequential and structured prediction problems (Collobert et al., 2011; Peng and Dredze, 2015).

Dropout Training. To mitigate overfitting, we apply the dropout method (Srivastava et al., 2014) to regularize our model. As shown in Figure 1 and 3, we apply dropout on character embeddings before inputting to CNN, and on both the input and output vectors of BLSTM. We fix dropout rate at 0.5 for all dropout layers through all the experiments. We obtain significant improvements on model performance after using dropout (see Section 4.5).

3.3 Tuning Hyper-Parameters

Table 1 summarizes the chosen hyper-parameters for all experiments. We tune the hyper-parameters on the development sets by random search. Due to time constrains it is infeasible to do a random search across the full hyper-parameter space. Thus, for the tasks of POS tagging and NER we try to share as many hyper-parameters as possible. Note that the final hyper-parameters for these two tasks are almost the same, except the initial learning rate. We set the state size of LSTM to 200. Tuning this parameter did not significantly impact the performance of our model. For CNN, we use 30 filters with window length 3.

4 Experiments

4.1 Data Sets

As mentioned before, we evaluate our neural network model on two sequence labeling tasks: POS tagging and NER.

Dataset		WSJ	CoNLL2003
Train	SENT	38,219	14,987
	TOKEN	912,344	204,567
Dev	SENT	5,527	3,466
	TOKEN	131,768	51,578
Test	SENT	5,462	3,684
	TOKEN	129,654	46,666

Table 2: Corpora statistics. SENT and TOKEN refer to the number of sentences and tokens in each data set.

POS Tagging. For English POS tagging, we use the Wall Street Journal (WSJ) portion of Penn Treebank (PTB) (Marcus et al., 1993), which contains 45 different POS tags. In order to compare with previous work, we adopt the standard splits — section 0–18 as training data, section 19–21 as development data and section 22–24 as test data (Manning, 2011; Søgaard, 2011).

NER. For NER, We perform experiments on the English data from CoNLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003). This data set contains four different types of named entities: *PERSON, LOCATION, ORGANIZATION*, and *MISC*. We use the BIOES tagging scheme instead of standard BIO2, as previous studies have reported meaningful improvement with this scheme (Ratinov and Roth, 2009; Dai et al., 2015; Lample et al., 2016).

The corpora statistics are shown in Table 2. We did not perform any pre-processing for data sets, leaving our system truly end-to-end.

4.2 Main Results

We first run experiments to dissect the effectiveness of each component (layer) of our neural network architecture by ablation studies. We compare the performance with three baseline systems — BRNN, the bi-direction RNN; BLSTM, the bi-direction LSTM, and BLSTM-CNNs, the combination of BLSTM with CNN to model character-level information. All these models are run using Stanford's GloVe 100 dimensional word embeddings and the same hyper-parameters as shown in Table 1. According to the results shown in Table 3, BLSTM obtains better performance than BRNN on all evaluation metrics of both the two tasks. BLSTM-CNN models significantly outperform the BLSTM model, showing that character-level representations are important for linguistic sequence labeling tasks. This is consistent with

	POS		NER					
	Dev	Test	Dev			Test		
Model	Acc.	Acc.	Prec.	Recall	F1	Prec.	Recall	F1
BRNN	96.56	96.76	92.04	89.13	90.56	87.05	83.88	85.44
BLSTM	96.88	96.93	92.31	90.85	91.57	87.77	86.23	87.00
BLSTM-CNN	97.34	97.33	92.52	93.64	93.07	88.53	90.21	89.36
BRNN-CNN-CRF	97.46	97.55	94.85	94.63	94.74	91.35	91.06	91.21

Table 3: Performance of our model on both the development and test sets of the two tasks, together with three baseline systems.

Model	**Acc.**
Giménez and Màrquez (2004)	97.16
Toutanova et al. (2003)	97.27
Manning (2011)	97.28
Collobert et al. (2011)‡	97.29
Santos and Zadrozny (2014)‡	97.32
Shen et al. (2007)	97.33
Sun (2014)	97.36
Søgaard (2011)	97.50
This paper	**97.55**

Table 4: POS tagging accuracy of our model on test data from WSJ proportion of PTB, together with top-performance systems. The neural network based models are marked with ‡.

Model	**F1**
Chieu and Ng (2002)	88.31
Florian et al. (2003)	88.76
Ando and Zhang (2005)	89.31
Collobert et al. (2011)‡	89.59
Huang et al. (2015)‡	90.10
Chiu and Nichols (2015)‡	90.77
Ratinov and Roth (2009)	90.80
Lin and Wu (2009)	90.90
Passos et al. (2014)	90.90
Lample et al. (2016)‡	90.94
Luo et al. (2015)	91.20
This paper	**91.21**

Table 5: NER F1 score of our model on test data set from CoNLL-2003. For the purpose of comparison, we also list F1 scores of previous top-performance systems. ‡ marks the neural models.

results reported by previous work (Santos and Zadrozny, 2014; Chiu and Nichols, 2015). Finally, by adding CRF layer for joint decoding we achieve significant improvements over BLSTM-CNN models for both POS tagging and NER on all metrics. This demonstrates that jointly decoding label sequences can significantly benefit the final performance of neural network models.

4.3 Comparison with Previous Work

4.3.1 POS Tagging

Table 4 illustrates the results of our model for POS tagging, together with seven previous top-performance systems for comparison. Our model significantly outperform Senna (Collobert et al., 2011), which is a feed-forward neural network model using capitalization and discrete suffix features, and data pre-processing. Moreover, our model achieves 0.23% improvements on accuracy over the "CharWNN" (Santos and Zadrozny, 2014), which is a neural network model based on Senna and also uses CNNs to model character-level representations. This demonstrates the effectiveness of BLSTM for modeling sequential data and the importance of joint decoding with structured prediction model.

Comparing with traditional statistical models, our system achieves state-of-the-art accuracy, obtaining 0.05% improvement over the previously best reported results by Søgaard (2011). It should be noted that Huang et al. (2015) also evaluated their BLSTM-CRF model for POS tagging on WSJ corpus. But they used a different splitting of the training/dev/test data sets. Thus, their results are not directly comparable with ours.

4.3.2 NER

Table 5 shows the F1 scores of previous models for NER on the test data set from CoNLL-2003 shared task. For the purpose of comparison, we list their results together with ours. Similar to the observations of POS tagging, our model achieves significant improvements over Senna and the other three neural models, namely the LSTM-CRF proposed by Huang et al. (2015), LSTM-CNNs pro-

Embedding	Dimension	POS	NER
Random	100	97.13	80.76
Senna	50	97.44	90.28
Word2Vec	300	97.40	84.91
GloVe	100	**97.55**	**91.21**

Table 6: Results with different choices of word embeddings on the two tasks (accuracy for POS tagging and F1 for NER).

posed by Chiu and Nichols (2015), and the LSTM-CRF by Lample et al. (2016). Huang et al. (2015) utilized discrete spelling, POS and context features, Chiu and Nichols (2015) used character-type, capitalization, and lexicon features, and all the three model used some task-specific data pre-processing, while our model does not require any carefully designed features or data pre-processing. We have to point out that the result (90.77%) reported by Chiu and Nichols (2015) is incomparable with ours, because their final model was trained on the combination of the training and development data sets[4].

To our knowledge, the previous best F1 score (91.20)[5] reported on CoNLL 2003 data set is by the joint NER and entity linking model (Luo et al., 2015). This model used many hand-crafted features including stemming and spelling features, POS and chunks tags, WordNet clusters, Brown Clusters, as well as external knowledge bases such as Freebase and Wikipedia. Our end-to-end model slightly improves this model by 0.01%, yielding a state-of-the-art performance.

4.4 Word Embeddings

As mentioned in Section 3.1, in order to test the importance of pretrained word embeddings, we performed experiments with different sets of publicly published word embeddings, as well as a random sampling method, to initialize our model. Table 6 gives the performance of three different word embeddings, as well as the randomly sampled one. According to the results in Table 6, models using pretrained word embeddings obtain a significant improvement as opposed to the ones using random embeddings. Comparing the two tasks, NER relies more heavily on pretrained embeddings than POS tagging. This is consistent with results reported by previous work (Collobert et al., 2011; Huang et al., 2015; Chiu and Nichols, 2015).

For different pretrained embeddings, Stanford's GloVe 100 dimensional embeddings achieve best results on both tasks, about 0.1% better on POS accuracy and 0.9% better on NER F1 score than the Senna 50 dimensional one. This is different from the results reported by Chiu and Nichols (2015), where Senna achieved slightly better performance on NER than other embeddings. Google's Word2Vec 300 dimensional embeddings obtain similar performance with Senna on POS tagging, still slightly behind GloVe. But for NER, the performance on Word2Vec is far behind GloVe and Senna. One possible reason that Word2Vec is not as good as the other two embeddings on NER is because of vocabulary mismatch — Word2Vec embeddings were trained in case-sensitive manner, excluding many common symbols such as punctuations and digits. Since we do not use any data pre-processing to deal with such common symbols or rare words, it might be an issue for using Word2Vec.

	POS			NER		
	Train	**Dev**	**Test**	**Train**	**Dev**	**Test**
No	98.46	97.06	97.11	99.97	93.51	89.25
Yes	97.86	97.46	97.55	99.63	94.74	91.21

Table 7: Results with and without dropout on two tasks (accuracy for POS tagging and F1 for NER).

	POS		NER	
	Dev	Test	Dev	Test
IV	127,247	125,826	4,616	3,773
OOTV	2,960	2,412	1,087	1,597
OOEV	659	588	44	8
OOBV	902	828	195	270

Table 8: Statistics of the partition on each corpus. It lists the number of tokens of each subset for POS tagging and the number of entities for NER.

4.5 Effect of Dropout

Table 7 compares the results with and without dropout layers for each data set. All other hyper-parameters remain the same as in Table 1. We observe a essential improvement for both the two tasks. It demonstrates the effectiveness of dropout in reducing overfitting.

[4]We run experiments using the same setting and get 91.37% F1 score.

[5]Numbers are taken from the Table 3 of the original paper (Luo et al., 2015). While there is clearly inconsistency among the precision (91.5%), recall (91.4%) and F1 scores (91.2%), it is unclear in which way they are incorrect.

	POS							
	Dev				Test			
	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV
LSTM-CNN	97.57	**93.75**	90.29	80.27	97.55	**93.45**	90.14	80.07
LSTM-CNN-CRF	**97.68**	93.65	**91.05**	**82.71**	**97.77**	93.16	**90.65**	**82.49**
	NER							
	Dev				Test			
	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV
LSTM-CNN	94.83	87.28	96.55	82.90	90.07	89.45	100.00	78.44
LSTM-CNN-CRF	**96.49**	**88.63**	**97.67**	**86.91**	**92.14**	**90.73**	100.00	**80.60**

Table 9: Comparison of performance on different subsets of words (accuracy for POS and F1 for NER).

4.6 OOV Error Analysis

To better understand the behavior of our model, we perform error analysis on Out-of-Vocabulary words (OOV). Specifically, we partition each data set into four subsets — in-vocabulary words (IV), out-of-training-vocabulary words (OOTV), out-of-embedding-vocabulary words (OOEV) and out-of-both-vocabulary words (OOBV). A word is considered IV if it appears in both the training and embedding vocabulary, while OOBV if neither. OOTV words are the ones do not appear in training set but in embedding vocabulary, while OOEV are the ones do not appear in embedding vocabulary but in training set. For NER, an entity is considered as OOBV if there exists at lease one word not in training set and at least one word not in embedding vocabulary, and the other three subsets can be done in similar manner. Table 8 informs the statistics of the partition on each corpus. The embedding we used is Stanford's GloVe with dimension 100, the same as Section 4.2.

Table 9 illustrates the performance of our model on different subsets of words, together with the baseline LSTM-CNN model for comparison. The largest improvements appear on the OOBV subsets of both the two corpora. This demonstrates that by adding CRF for joint decoding, our model is more powerful on words that are out of both the training and embedding sets.

5 Related Work

In recent years, several different neural network architectures have been proposed and successfully applied to linguistic sequence labeling such as POS tagging, chunking and NER. Among these neural architectures, the three approaches most similar to our model are the BLSTM-CRF model proposed by Huang et al. (2015), the LSTM-CNNs model by Chiu and Nichols (2015) and the BLSTM-CRF by Lample et al. (2016).

Huang et al. (2015) used BLSTM for word-level representations and CRF for jointly label decoding, which is similar to our model. But there are two main differences between their model and ours. First, they did not employ CNNs to model character-level information. Second, they combined their neural network model with handcrafted features to improve their performance, making their model not an end-to-end system. Chiu and Nichols (2015) proposed a hybrid of BLSTM and CNNs to model both character- and word-level representations, which is similar to the first two layers in our model. They evaluated their model on NER and achieved competitive performance. Our model mainly differ from this model by using CRF for joint decoding. Moreover, their model is not truly end-to-end, either, as it utilizes external knowledge such as character-type, capitalization and lexicon features, and some data preprocessing specifically for NER (e.g. replacing all sequences of digits 0-9 with a single "0"). Recently, Lample et al. (2016) proposed a BLSTM-CRF model for NER, which utilized BLSTM to model both the character- and word-level information, and use data pre-processing the same as Chiu and Nichols (2015). Instead, we use CNN to model character-level information, achieving better NER performance without using any data preprocessing.

There are several other neural networks previously proposed for sequence labeling. Labeau et al. (2015) proposed a RNN-CNNs model for German POS tagging. This model is similar to the LSTM-CNNs model in Chiu and Nichols (2015), with the difference of using vanila RNN instead of LSTM. Another neural architecture employing

CNN to model character-level information is the "CharWNN" architecture (Santos and Zadrozny, 2014) which is inspired by the feed-forward network (Collobert et al., 2011). CharWNN obtained near state-of-the-art accuracy on English POS tagging (see Section 4.3 for details). A similar model has also been applied to Spanish and Portuguese NER (dos Santos et al., 2015) Ling et al. (2015) and Yang et al. (2016) also used BSLTM to compose character embeddings to word's representation, which is similar to Lample et al. (2016). Peng and Dredze (2016) Improved NER for Chinese Social Media with Word Segmentation.

6 Conclusion

In this paper, we proposed a neural network architecture for sequence labeling. It is a truly end-to-end model relying on no task-specific resources, feature engineering or data pre-processing. We achieved state-of-the-art performance on two linguistic sequence labeling tasks, comparing with previously state-of-the-art systems.

There are several potential directions for future work. First, our model can be further improved by exploring multi-task learning approaches to combine more useful and correlated information. For example, we can jointly train a neural network model with both the POS and NER tags to improve the intermediate representations learned in our network. Another interesting direction is to apply our model to data from other domains such as social media (Twitter and Weibo). Since our model does not require any domain- or task-specific knowledge, it might be effortless to apply it to these domains.

Acknowledgements

This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

[Ando and Zhang2005] Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853.

[Bengio et al.1994] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

[Bergstra et al.2010] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3. Austin, TX.

[Chen and Manning2014] Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP-2014*, pages 740–750, Doha, Qatar, October.

[Chieu and Ng2002] Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of CoNLL-2003*, pages 1–7.

[Chiu and Nichols2015] Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

[Cho et al.2014] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103.

[Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

[Dai et al.2015] Hong-Jie Dai, Po-Ting Lai, Yung-Chun Chang, and Richard Tzong-Han Tsai. 2015. Enhancing of chemical compound and drug name recognition using representative tag scheme and fine-grained tokenization. *Journal of cheminformatics*, 7(S1):1–10.

[Dauphin et al.2015] Yann N Dauphin, Harm de Vries, Junyoung Chung, and Yoshua Bengio. 2015. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *arXiv preprint arXiv:1502.04390*.

[dos Santos et al.2015] Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 25.

[Dyer et al.2015] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL-2015 (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July.

[Florian et al.2003] Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of HLT-NAACL-2003*, pages 168–171.

[Gers et al.2000] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471.

[Gers et al.2003] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. 2003. Learning precise timing with lstm recurrent networks. *The Journal of Machine Learning Research*, 3:115–143.

[Giles2001] Rich Caruana Steve Lawrence Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, volume 13, page 402. MIT Press.

[Giménez and Màrquez2004] Jesús Giménez and Lluís Màrquez. 2004. Svmtool: A general pos tagger generator based on support vector machines. In *In Proceedings of LREC-2004*.

[Glorot and Bengio2010] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

[Goller and Kuchler1996] Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE.

[Graves et al.2013] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP-2013*, pages 6645–6649. IEEE.

[He et al.2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hu et al.2016] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard H. Hovy, and Eric P. Xing. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of ACL-2016*, Berlin, Germany, August.

[Huang et al.2015] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

[Jozefowicz et al.2015] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350.

[Kingma and Ba2014] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Koo and Collins2010] Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL-2010*, pages 1–11, Uppsala, Sweden, July.

[Labeau et al.2015] Matthieu Labeau, Kevin Löser, Alexandre Allauzen, and Rue John von Neumann. 2015. Non-lexical neural architecture for fine-grained pos tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 232–237.

[Lafferty et al.2001] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-2001*, volume 951, pages 282–289.

[Lample et al.2016] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-2016*, San Diego, California, USA, June.

[LeCun et al.1989] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[Lin and Wu2009] Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of ACL-2009*, pages 1030–1038.

[Ling et al.2015] Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of EMNLP-2015*, pages 1520–1530, Lisbon, Portugal, September.

[Luo et al.2015] Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of EMNLP-2015*, pages 879–888, Lisbon, Portugal, September.

[Ma and Hovy2015] Xuezhe Ma and Eduard Hovy. 2015. Efficient inner-to-outer greedy algorithm for higher-order labeled dependency parsing. In *Proceedings of the EMNLP-2015*, pages 1322–1328, Lisbon, Portugal, September.

[Ma and Xia2014] Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of ACL-2014*, pages 1337–1348, Baltimore, Maryland, June.

[Ma and Zhao2012a] Xuezhe Ma and Hai Zhao. 2012a. Fourth-order dependency parsing. In *Proceedings of COLING 2012: Posters*, pages 785–796, Mumbai, India, December.

[Ma and Zhao2012b] Xuezhe Ma and Hai Zhao. 2012b. Probabilistic models for high-order projective dependency parsing. *Technical Report, arXiv:1502.04174*.

[Ma et al.2016] Xuezhe Ma, Zhengzhong Liu, and Eduard Hovy. 2016. Unsupervised ranking model for entity coreference resolution. In *Proceedings of NAACL-2016*, San Diego, California, USA, June.

[Manning2011] Christopher D Manning. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*, pages 171–189. Springer.

[Marcus et al.1993] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

[McDonald et al.2005] Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL-2005*, pages 91–98, Ann Arbor, Michigan, USA, June 25-30.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Ng2010] Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of ACL-2010*, pages 1396–1411, Uppsala, Sweden, July. Association for Computational Linguistics.

[Nivre and Scholz2004] Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING-2004*, pages 64–70, Geneva, Switzerland, August 23-27.

[Pascanu et al.2012] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

[Passos et al.2014] Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of CoNLL-2014*, pages 78–86, Ann Arbor, Michigan, June.

[Peng and Dredze2015] Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of EMNLP-2015*, pages 548–554, Lisbon, Portugal, September.

[Peng and Dredze2016] Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *Proceedings of ACL-2016*, Berlin, Germany, August.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP-2014*, pages 1532–1543, Doha, Qatar, October.

[Ratinov and Roth2009] Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL-2009*, pages 147–155.

[Santos and Zadrozny2014] Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of ICML-2014*, pages 1818–1826.

[Shen et al.2007] Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of ACL-2007*, volume 7, pages 760–767.

[Søgaard2011] Anders Søgaard. 2011. Semi-supervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 48–52, Portland, Oregon, USA, June.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[Sun2014] Xu Sun. 2014. Structure regularization for structured prediction. In *Advances in Neural Information Processing Systems*, pages 2402–2410.

[Tjong Kim Sang and De Meulder2003] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003 - Volume 4*, pages 142–147, Stroudsburg, PA, USA.

[Tjong Kim Sang and Veenstra1999] Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99*, pages 173–179. Bergen, Norway.

[Toutanova et al.2003] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003.

Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of NAACL-HLT-2003, Volume 1*, pages 173–180.

[Yang et al.2016] Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

[Zeiler2012] Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Strong Baselines for Neural Semi-supervised Learning under Domain Shift

Sebastian Ruder[♠♣] **Barbara Plank**[♡◇]
[♠]Insight Research Centre, National University of Ireland, Galway, Ireland
[♣]Aylien Ltd., Dublin, Ireland
[♡]Center for Language and Cognition, University of Groningen, The Netherlands
[◇]Department of Computer Science, IT University of Copenhagen, Denmark
sebastian@ruder.io,bplank@gmail.com

Abstract

Novel neural models have been proposed in recent years for learning under domain shift. Most models, however, only evaluate on a single task, on proprietary datasets, or compare to weak baselines, which makes comparison of models difficult. In this paper, we re-evaluate classic general-purpose bootstrapping approaches in the context of neural networks under domain shifts vs. recent neural approaches and propose a novel *multi-task tri-training* method that reduces the time and space complexity of classic tri-training. Extensive experiments on two benchmarks are negative: while our novel method establishes a new state-of-the-art for sentiment analysis, it does not fare consistently the best. More importantly, we arrive at the somewhat surprising conclusion that classic tri-training, with some additions, outperforms the state of the art. We conclude that classic approaches constitute an important and strong baseline.

1 Introduction

Deep neural networks (DNNs) excel at learning from labeled data and have achieved state of the art in a wide array of supervised NLP tasks such as dependency parsing (Dozat and Manning, 2017), named entity recognition (Lample et al., 2016), and semantic role labeling (He et al., 2017).

In contrast, learning from unlabeled data, especially under domain shift, remains a challenge. This is common in many real-world applications where the distribution of the training and test data differs. Many state-of-the-art domain adaptation approaches leverage task-specific characteristics such as sentiment words (Blitzer et al., 2006; Wu and Huang, 2016) or distributional features (Schnabel and Schütze, 2014; Yin et al., 2015) which do not generalize to other tasks. Other approaches that are in theory more general only evaluate on proprietary datasets (Kim et al., 2017) or on a single benchmark (Zhou et al., 2016), which carries the risk of overfitting to the task. In addition, most models only compare against weak baselines and, strikingly, almost none considers evaluating against approaches from the extensive semi-supervised learning (SSL) literature (Chapelle et al., 2006).

In this work, we make the argument that such algorithms make strong baselines for any task in line with recent efforts highlighting the usefulness of classic approaches (Melis et al., 2017; Denkowski and Neubig, 2017). We re-evaluate bootstrapping algorithms in the context of DNNs. These are general-purpose semi-supervised algorithms that treat the model as a black box and can thus be used easily—with a few additions—with the current generation of NLP models. Many of these methods, though, were originally developed with in-domain performance in mind, so their effectiveness in a domain adaptation setting remains unexplored.

In particular, we re-evaluate three traditional bootstrapping methods, self-training (Yarowsky, 1995), tri-training (Zhou and Li, 2005), and tri-training with disagreement (Søgaard, 2010) for neural network-based approaches on *two* NLP tasks with different characteristics, namely, a sequence prediction and a classification task (POS tagging and sentiment analysis). We evaluate the methods across multiple domains on two well-established benchmarks, without taking any further task-specific measures, and compare to the best results published in the literature.

We make the somewhat surprising observation that classic tri-training outperforms task-agnostic state-of-the-art semi-supervised learning (Laine and Aila, 2017) and recent neural adaptation approaches (Ganin et al., 2016; Saito et al., 2017).

In addition, we propose *multi-task tri-training*, which reduces the main deficiency of tri-training, namely its time and space complexity. It establishes a new state of the art on unsupervised domain adaptation for sentiment analysis but it is outperformed by classic tri-training for POS tagging.

Contributions Our contributions are: a) We propose a novel multi-task tri-training method. b) We show that tri-training can serve as a strong and robust semi-supervised learning baseline for the current generation of NLP models. c) We perform an extensive evaluation of bootstrapping[1] algorithms compared to state-of-the-art approaches on two benchmark datasets. d) We shed light on the task and data characteristics that yield the best performance for each model.

2 Neural bootstrapping methods

We first introduce three classic bootstrapping methods, self-training, tri-training, and tri-training with disagreement and detail how they can be used with neural networks. For in-depth details we refer the reader to (Abney, 2007; Chapelle et al., 2006; Zhu and Goldberg, 2009). We introduce our novel multi-task tri-training method in §2.3.

2.1 Self-training

Self-training (Yarowsky, 1995; McClosky et al., 2006b) is one of the earliest and simplest bootstrapping approaches. In essence, it leverages the model's own predictions on unlabeled data to obtain additional information that can be used during training. Typically the most confident predictions are taken at face value, as detailed next.

Self-training trains a model m on a labeled training set L and an unlabeled data set U. At each iteration, the model provides predictions $m(x)$ in the form of a probability distribution over classes for all unlabeled examples x in U. If the probability assigned to the most likely class is higher than a predetermined threshold τ, x is added to the labeled examples with $p(x) = \arg\max m(x)$ as pseudo-label. This instantiation is the most widely used and shown in Algorithm 1.

Calibration It is well-known that output probabilities in neural networks are poorly calibrated (Guo et al., 2017). Using a fixed threshold τ is thus not the best choice. While the *absolute* confidence value is inaccurate, we can expect that the *relative* order of confidences is more robust.

For this reason, we select the top n unlabeled examples that have been predicted with the highest confidence after every epoch and add them to the labeled data. This is one of the many variants for self-training, called *throttling* (Abney, 2007). We empirically confirm that this outperforms the classic selection in our experiments.

Algorithm 1 Self-training (Abney, 2007)

repeat
  $m \leftarrow train\_model(L)$
  for $x \in U$ **do**
    if $\max m(x) > \tau$ **then**
      $L \leftarrow L \cup \{(x, p(x))\}$
until no more predictions are confident

Online learning In contrast to many classic algorithms, DNNs are trained online by default. We compare training setups and find that training until convergence on labeled data and then training until convergence using self-training performs best.

Classic self-training has shown mixed success. In parsing it proved successful only with small datasets (Reichart and Rappoport, 2007) or when a generative component is used together with a reranker in high-data conditions (McClosky et al., 2006b; Suzuki and Isozaki, 2008). Some success was achieved with careful task-specific data selection (Petrov and McDonald, 2012), while others report limited success on a variety of NLP tasks (Plank, 2011; Van Asch and Daelemans, 2016; van der Goot et al., 2017). Its main downside is that the model is not able to correct its own mistakes and errors are amplified, an effect that is increased under domain shift.

2.2 Tri-training

Tri-training (Zhou and Li, 2005) is a classic method that reduces the bias of predictions on unlabeled data by utilizing the agreement of three independently trained models. Tri-training (cf. Algorithm 2) first trains three models m_1, m_2, and m_3 on bootstrap samples of the labeled data L. An unlabeled data point is added to the training set of a model m_i if the other two models m_j and m_k agree on its label. Training stops when the classifiers do not change anymore.

Tri-training *with disagreement* (Søgaard, 2010)

[1] We use the term bootstrapping as used in the semi-supervised learning literature (Zhu, 2005), which should not be confused with the statistical procedure of the same name (Efron and Tibshirani, 1994).

Algorithm 2 Tri-training (Zhou and Li, 2005)

1: **for** $i \in \{1..3\}$ **do**
2: $S_i \leftarrow bootstrap\_sample(L)$
3: $m_i \leftarrow train\_model(S_i)$
4: **repeat**
5: **for** $i \in \{1..3\}$ **do**
6: $L_i \leftarrow \emptyset$
7: **for** $x \in U$ **do**
8: **if** $p_j(x) = p_k(x)(j, k \neq i)$ **then**
9: $L_i \leftarrow L_i \cup \{(x, p_j(x))\}$
$m_i \leftarrow train\_model(L \cup L_i)$
10: **until** none of m_i changes
11: apply majority vote over m_i

is based on the intuition that a model should only be strengthened in its weak points and that the labeled data should not be skewed by easy data points. In order to achieve this, it adds a simple modification to the original algorithm (altering line 8 in Algorithm 2), requiring that for an unlabeled data point on which m_j and m_k *agree*, the other model m_i *disagrees* on the prediction. Tri-training with disagreement is more data-efficient than tri-training and has achieved competitive results on part-of-speech tagging (Søgaard, 2010).

Sampling unlabeled data Both tri-training and tri-training with disagreement can be very expensive in their original formulation as they require to produce predictions for each of the three models on all unlabeled data samples, which can be in the millions in realistic applications. We thus propose to sample a number of unlabeled examples at every epoch. For all traditional bootstrapping approaches we sample 10k candidate instances in each epoch. For the neural approaches we use a linearly growing candidate sampling scheme proposed by (Saito et al., 2017), increasing the candidate pool size as the models become more accurate.

Confidence thresholding Similar to self-training, we can introduce an additional requirement that pseudo-labeled examples are only added if the probability of the prediction of at least one model is higher than some threshold τ. We did not find this to outperform prediction without threshold for traditional tri-training, but thresholding proved essential for our method (§2.3).

The most important condition for tri-training and tri-training with disagreement is that the models are diverse. Typically, bootstrap samples are used to create this diversity (Zhou and Li, 2005; Søgaard, 2010). However, training separate models on bootstrap samples of a potentially large amount of training data is expensive and takes a lot of time. This drawback motivates our approach.

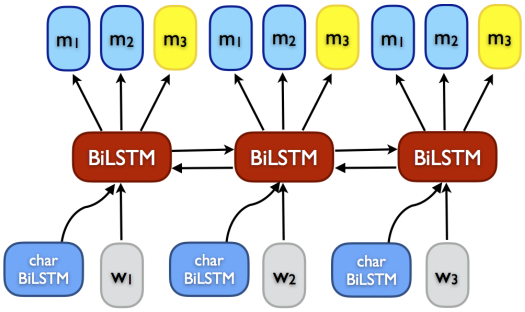


Figure 1: Multi-task tri-training (MT-Tri).

2.3 Multi-task tri-training

In order to reduce both the time and space complexity of tri-training, we propose Multi-task Tri-training (MT-Tri). MT-Tri leverages insights from multi-task learning (MTL) (Caruana, 1993) to share knowledge across models and accelerate training. Rather than storing and training each model separately, we propose to share the parameters of the models and train them jointly using MTL.[2] All models thus collaborate on learning a joint representation, which improves convergence.

The output softmax layers are model-specific and are only updated for the input of the respective model. We show the model in Figure 1 (as instantiated for POS tagging). As the models leverage a joint representation, we need to ensure that the features used for prediction in the softmax layers of the different models are as diverse as possible, so that the models can still learn from each other's predictions. In contrast, if the parameters in all output softmax layers were the same, the method would degenerate to self-training.

To guarantee diversity, we introduce an orthogonality constraint (Bousmalis et al., 2016) as an additional loss term, which we define as follows:

$$\mathcal{L}_{orth} = \|W_{m_1}^\top W_{m_2}\|_F^2 \quad (1)$$

where $\|\cdot\|_F^2$ is the squared Frobenius norm and W_{m_1} and W_{m_2} are the softmax output parameters

[2]Note: we use the term multi-task learning here albeit all tasks are of the same kind, similar to work on multi-lingual modeling treating each language (but same label space) as separate task e.g., (Fang and Cohn, 2017). It is interesting to point out that our model is further doing implicit multi-view learning by way of the orthogonality constraint.

of the two source and pseudo-labeled output layers m_1 and m_2, respectively. The orthogonality constraint encourages the models not to rely on the same features for prediction. As enforcing pair-wise orthogonality between three matrices is not possible, we only enforce orthogonality between the softmax output layers of m_1 and m_2,[3] while m_3 is gradually trained to be more target-specific. We parameterize $\mathcal{L}_{orth}$ by γ=0.01 following (Liu et al., 2017). We do not further tune γ.

More formally, let us illustrate the model by taking the sequence prediction task (Figure 1) as illustration. Given an utterance with labels $y_1, .., y_n$, our Multi-task Tri-training loss consists of three task-specific (m_1, m_2, m_3) tagging loss functions (where $\vec{h}$ is the uppermost Bi-LSTM encoding):

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i}\sum_{1,..,n} \log P_{m_i}(y|\vec{h}) + \gamma\mathcal{L}_{orth} \quad (2)$$

In contrast to classic tri-training, we can train the multi-task model with its three model-specific outputs jointly and *without* bootstrap sampling on the labeled source domain data until convergence, as the orthogonality constraint enforces different representations between models m_1 and m_2. From this point, we can leverage the pair-wise agreement of two output layers to add pseudo-labeled examples as training data to the third model. We train the third output layer m_3 only on pseudo-labeled target instances in order to make tri-training more robust to a domain shift. For the final prediction, majority voting of all three output layers is used, which resulted in the best instantiation, together with confidence thresholding ($\tau = 0.9$, except for high-resource POS where $\tau = 0.8$ performed slightly better). We also experimented with using a domain-adversarial loss (Ganin et al., 2016) on the jointly learned representation, but found this not to help. The full pseudo-code is given in Algorithm 3.

Computational complexity The motivation for MT-Tri was to reduce the space and time complexity of tri-training. We thus give an estimate of its efficiency gains. MT-Tri is ~3× more space-efficient than regular tri-training; tri-training stores one set of parameters for each of the three models, while MT-Tri only stores one set of parameters (we use three output layers, but these make up a comparatively small part of the total parameter budget). In terms of time efficiency, tri-training first requires to train each of the models from scratch. The actual tri-training takes about the same time as training from scratch and requires a separate forward pass for each model, effectively training three independent models simultaneously. In contrast, MT-Tri only necessitates one forward pass as well as the evaluation of the two additional output layers (which takes a negligible amount of time) and requires about as many epochs as tri-training until convergence (see Table 3, second column) while adding fewer unlabeled examples per epoch (see Section 3.4). In our experiments, MT-Tri trained about 5-6× faster than traditional tri-training.

Algorithm 3 Multi-task Tri-training

$m \leftarrow train\_model(L)$
repeat
  for $i \in \{1..3\}$ **do**
    $L_i \leftarrow \emptyset$
    for $x \in U$ **do**
      if $p_j(x) = p_k(x)(j, k \neq i)$ **then**
        $L_i \leftarrow L_i \cup \{(x, p_j(x))\}$
    if $i = 3$ **then** $m_i = train\_model(L_i)$
    else $m_i \leftarrow train\_model(L \cup L_i)$
until end condition is met
apply majority vote over m_i

MT-Tri can be seen as a self-ensembling technique, where different variations of a model are used to create a stronger ensemble prediction. Recent approaches in this line are *snapshot ensembling* (Huang et al., 2017) that ensembles models converged to different minima during a training run, *asymmetric tri-training* (Saito et al., 2017) (ASYM) that leverages agreement on two models as information for the third, and *temporal ensembling* (Laine and Aila, 2017), which ensembles predictions of a model at different epochs. We tried to compare to temporal ensembling in our experiments, but were not able to obtain consistent results.[4] We compare to the closest most recent method, asymmetric tri-training (Saito et al., 2017). It differs from ours in two aspects: a) ASYM leverages only pseudo-labels from data points on which m_1 and m_2 agree, and b) it uses only one task (m_3) as final predictor. In essence, our formulation of MT-Tri is closer to the original tri-training formulation (agreements on two provide pseudo-labels to the third) thereby incorporating more diversity.

[3] We also tried enforcing orthogonality on a hidden layer rather than the output layer, but this did not help.

[4] We suspect that the sparse features in NLP and the domain shift might be detrimental to its unsupervised consistency loss.

	Domain	# labeled	# unlabeled
POS tagging	Answers	3,489	27,274
	Emails	4,900	1,194,173
	Newsgroups	2,391	1,000,000
	Reviews	3,813	1,965,350
	Weblogs	2,031	524,834
	WSJ	30,060	100,000
Sentiment	Book	2,000	4,465
	DVD	2,000	3,586
	Electronics	2,000	5,681
	Kitchen	2,000	5,945

Table 1: Number of labeled and unlabeled sentences for each domain in the SANCL 2012 dataset (Petrov and McDonald, 2012) for POS tagging (above) and the Amazon Reviews dataset (Blitzer et al., 2006) for sentiment analysis (below).

3 Experiments

In order to ascertain which methods are robust across different domains, we evaluate on two widely used unsupervised domain adaptation datasets for two tasks, a sequence labeling and a classification task, cf. Table 1 for data statistics.

3.1 POS tagging

For POS tagging we use the SANCL 2012 shared task dataset (Petrov and McDonald, 2012) and compare to the top results in both low and high-data conditions (Schnabel and Schütze, 2014; Yin et al., 2015). Both are strong baselines, as the FLORS tagger has been developed for this challenging dataset and it is based on contextual distributional features (excluding the word's identity), and hand-crafted suffix and shape features (including some language-specific morphological features). We want to gauge to what extent we can adopt a nowadays fairly standard (but more lexicalized) general neural tagger.

Our POS tagging model is a state-of-the-art Bi-LSTM tagger (Plank et al., 2016) with word and 100-dim character embeddings. Word embeddings are initialized with the 100-dim Glove embeddings (Pennington et al., 2014). The BiLSTM has one hidden layer with 100 dimensions. The base POS model is trained on WSJ with early stopping on the WSJ development set, using patience 2, Gaussian noise with $\sigma = 0.2$ and word dropout with $p = 0.25$ (Kiperwasser and Goldberg, 2016).

Regarding data, the source domain is the Ontonotes 4.0 release of the Penn treebank Wall Street Journal (WSJ) annotated for 48 fine-grained POS tags. This amounts to 30,060 labeled sentences. We use 100,000 WSJ sentences from 1988 as unlabeled data, following Schnabel and Schütze (2014).[5] As target data, we use the five SANCL domains (answers, emails, newsgroups, reviews, weblogs). We restrict the amount of unlabeled data for each SANCL domain to the first 100k sentences, and do not do any pre-processing. We consider the development set of ANSWERS as our only target dev set to set hyperparameters. This may result in suboptimal per-domain settings but better resembles an unsupervised adaptation scenario.

3.2 Sentiment analysis

For sentiment analysis, we evaluate on the Amazon reviews dataset (Blitzer et al., 2006). Reviews with 1 to 3 stars are ranked as negative, while reviews with 4 or 5 stars are ranked as positive. The dataset consists of four domains, yielding 12 adaptation scenarios. We use the same pre-processing and architecture as used in (Ganin et al., 2016; Saito et al., 2017): 5,000-dimensional tf-idf weighted unigram and bigram features as input; 2k labeled source samples and 2k unlabeled target samples for training, 200 labeled target samples for validation, and between 3k-6k samples for testing. The model is an MLP with one hidden layer with 50 dimensions, sigmoid activations, and a softmax output. We compare against the Variational Fair Autoencoder (VFAE) (Louizos et al., 2015) model and domain-adversarial neural networks (DANN) (Ganin et al., 2016).

3.3 Baselines

Besides comparing to the top results published on both datasets, we include the following baselines:

a) the task model trained on the source domain;
b) self-training (Self);
c) tri-training (Tri);
d) tri-training with disagreement (Tri-D); and
e) asymmetric tri-training (Saito et al., 2017).

Our proposed model is multi-task tri-training (MT-Tri). We implement our models in DyNet (Neubig et al., 2017). Reporting single evaluation scores might result in biased results (Reimers and Gurevych, 2017). Throughout the paper, we report mean accuracy and standard deviation over five runs for POS tagging and over ten runs for

[5] Note that our unlabeled data might slightly differ from theirs. We took the first 100k sentences from the 1988 WSJ dataset from the BLLIP 1987-89 WSJ Corpus Release 1.

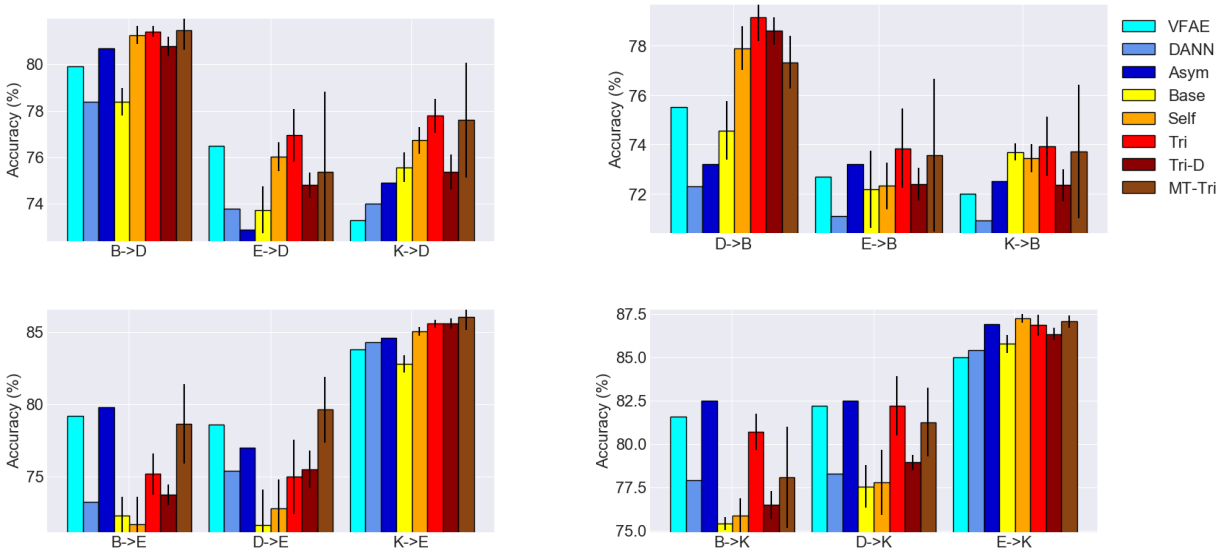


Figure 2: Average results for unsupervised domain adaptation on the Amazon dataset. Domains: B (Book), D (DVD), E (Electronics), K (Kitchen). Results for VFAE, DANN, and Asym are from Saito et al. (2017).

sentiment analysis. Significance is computed using bootstrap test. The code for all experiments is released at: https://github.com/bplank/semi-supervised-baselines.

3.4 Results

Sentiment analysis We show results for sentiment analysis for all 12 domain adaptation scenarios in Figure 2. For clarity, we also show the accuracy scores averaged across each target domain as well as a global macro average in Table 2.

Model	D	B	E	K	Avg
VFAE*	76.57	73.40	80.53	82.93	78.36
DANN*	75.40	71.43	77.67	80.53	76.26
Asym*	76.17	72.97	80.47	**83.97**	78.39
Src	75.91	73.47	75.61	79.58	76.14
Self	78.00	74.55	76.54	80.30	77.35
Tri	**78.72**	**75.64**	78.60	83.26	79.05
Tri-D	76.99	74.44	78.30	80.59	77.58
MT-Tri	78.14	74.86	**81.45**	82.14	**79.15**

Table 2: Average accuracy scores for each SA target domain. *: result from Saito et al. (2017).

Self-training achieves surprisingly good results but is not able to compete with tri-training. Tri-training with disagreement is only slightly better than self-training, showing that the disagreement component might not be useful when there is a strong domain shift. Tri-training achieves the best average results on two target domains and clearly outperforms the state of the art on average.

MT-Tri finally outperforms the state of the art on 3/4 domains, and even slightly traditional tri-training, resulting in the overall best method. This improvement is mainly due to the B->E and D->E scenarios, on which tri-training struggles. These domain pairs are among those with the highest $\mathcal{A}$-distance (Blitzer et al., 2007), which highlights that tri-training has difficulty dealing with a strong shift in domain. Our method is able to mitigate this deficiency by training one of the three output layers only on pseudo-labeled target domain examples.

In addition, MT-Tri is more efficient as it adds a smaller number of pseudo-labeled examples than tri-training at every epoch. For sentiment analysis, tri-training adds around 1800-1950/2000 unlabeled examples at every epoch, while MT-Tri only adds around 100-300 in early epochs. This shows that the orthogonality constraint is useful for inducing diversity. In addition, adding fewer examples poses a smaller risk of swamping the learned representations with useless signals and is more akin to fine-tuning, the standard method for supervised domain adaptation (Howard and Ruder, 2018).

We observe an asymmetry in the results between some of the domain pairs, e.g. B->D and D->B. We hypothesize that the asymmetry may be due to properties of the data and that the domains are relatively far apart e.g., in terms of $\mathcal{A}$-distance. In fact, asymmetry in these domains is already reflected

Model	ep	Target domains: Answers	Emails	Newsgroups	Reviews	Weblogs	Avg	WSJ	μ_{pseudo}
Src (+glove)		87.63 ±.37	86.49 ±.35	**88.60** ±.22	90.12 ±.32	92.85 ±.17	89.14 ±.28	95.49 ±.09	—
Self	(5)	87.64 ±.18	86.58 ±.30	88.42 ±.24	90.03 ±.11	92.80 ±.19	89.09 ±.20	95.36 ±.07	.5k
Tri	(4)	88.42 ±.16	87.46 ±.20	87.97 ±.09	90.72 ±.14	93.40 ±.15	89.56 ±.16	95.94 ±.07	20.5k
Tri-D	(7)	**88.50** ±.04	**87.63** ±.15	88.12 ±.05	**90.76** ±.10	**93.51** ±.06	**89.70** ±.08	**95.99** ±.03	7.7K
Asym	(3)	87.81 ±.19	86.97 ±.17	87.74 ±.24	90.16 ±.17	92.73 ±.16	89.08 ±.19	95.55 ±.12	1.5k
MT-Tri	(4)	87.92 ±.18	87.20 ±.23	87.73 ±.37	90.27 ±.10	92.96 ±.07	89.21 ±.19	95.50 ±.06	7.6k
FLORS		89.71	88.46	89.82	92.10	94.20	90.86	95.80	—

Table 3: Accuracy scores on dev set of target domain for POS tagging for 10% labeled data. Avg: average over the 5 SANCL domains. Hyperparameter ep (epochs) is tuned on Answers dev. μ_{pseudo}: average amount of added pseudo-labeled data. FLORS: results for Batch (u:big) from (Yin et al., 2015) (see §3).

Model	Target domains dev sets: Answers	Emails	Newsgroups	Reviews	Weblogs	Avg on targets	WSJ
TnT*	88.55	88.14	88.66	90.40	93.33	89.82	95.75
Stanford*	88.92	88.68	89.11	91.43	94.15	90.46	96.83
Src	88.84 ±.15	88.24 ±.12	89.45 ±.23	91.24 ±.03	93.92 ±.17	90.34 ±.14	96.69 ±.08
Tri	89.34 ±.18	88.83 ±.07	89.32 ±.21	91.62 ±.06	94.40 ±.06	90.70 ±.12	96.84 ±.04
Tri-D	89.35 ±.16	88.66 ±.09	89.29 ±.12	91.58 ±.05	94.32 ±.05	90.62 ±.09	96.85 ±.06
Src (+glove)	89.35 ±.16	88.55 ±.14	**90.12** ±.31	91.48 ±.15	94.48 ±.07	90.80 ±.17	96.90 ±.04
Tri	**90.00** ±.03	**89.06** ±.16	90.04 ±.25	**91.98** ±.11	**94.74** ±.06	**91.16** ±.12	**96.99** ±.02
Tri-D	89.80 ±.19	88.85 ±.10	90.03 ±.22	**91.98** ±.09	94.70 ±.05	91.01 ±.13	96.95 ±.05
Asym	89.51 ±.15	88.47 ±.19	89.26 ±.16	91.60 ±.20	94.28 ±.15	90.62 ±.17	96.56 ±.01
MT-Tri	89.45 ±.05	88.65 ±.04	89.40 ±.22	91.63 ±.23	94.41 ±.05	90.71 ±.12	97.37 ±.07
FLORS*	90.30	89.44	90.86	92.95	94.71	91.66	96.59
Model	**Target domains test sets: Answers**	**Emails**	**Newsgroups**	**Reviews**	**Weblogs**	**Avg on targets**	**WSJ**
TnT*	89.36	87.38	90.85	89.67	91.37	89.73	96.57
Stanford*	89.74	87.77	91.25	90.30	92.32	90.28	97.43
Src (+glove)	90.43 ±.13	87.95 ±.18	91.83 ±.20	90.04 ±.11	92.44 ±.14	90.54 ±.15	**97.50** ±.03
Tri	**91.21** ±.06	**88.30** ±.19	**92.18** ±.19	**90.06** ±.10	**92.85** ±.02	**90.92** ±.11	97.45 ±.03
Asym	90.62 ±.26	87.71 ±.07	91.40 ±.05	89.89 ±.22	92.37 ±.27	90.39 ±.17	97.19 ±.03
MT-Tri	90.53 ±.15	87.90 ±.07	91.45 ±.19	89.77 ±.26	92.35 ±.09	90.40 ±.15	97.37 ±.07
FLORS*	91.17	88.67	92.41	92.25	93.14	91.53	97.11

Table 4: Accuracy for POS tagging on the dev and test sets of the SANCL domains, models trained on full source data setup. Values for methods with * are from (Schnabel and Schütze, 2014).

in the results of Blitzer et al. (2007) and is corroborated in the results for asymmetric tri-training (Saito et al., 2017) and our method.

We note a weakness of this dataset is high variance. Existing approaches only report the mean, which makes an objective comparison difficult. For this reason, we believe it is essential to evaluate proposed approaches also on other tasks.

POS tagging Results for tagging in the low-data regime (10% of WSJ) are given in Table 3.

Self-training does not work for the sequence prediction task. We report only the best instantiation (throttling with n=800). Our results contribute to negative findings regarding self-training (Plank, 2011; Van Asch and Daelemans, 2016).

In the low-data setup, tri-training *with disagreement* works best, reaching an overall average accuracy of 89.70, closely followed by classic tri-training, and significantly outperforming the baseline on 4/5 domains. The exception is newsgroups, a difficult domain with high OOV rate where none of the approches beats the baseline (see §3.4). Our proposed MT-Tri is better than asymmetric tri-training, but falls below classic tri-training. It beats

	Ans	Email	Newsg	Rev	Webl
% unk tag	0.25	0.80	0.31	0.06	0.0
% OOV	8.53	10.56	10.34	6.84	8.45
% UWT	2.91	3.47	2.43	2.21	1.46
Accuracy on OOV tokens					
Src	54.26	57.48	**61.80**	59.26	**80.37**
Tri	**55.53**	**59.11**	61.36	**61.16**	79.32
Asym	52.86	56.78	56.58	59.59	76.84
MT-Tri	52.88	57.22	57.28	58.99	77.77
Accuracy on unknown word-tag (UWT) tokens					
Src	**17.68**	**11.14**	**17.88**	**17.31**	**24.79**
Tri	16.88	10.04	17.58	16.35	23.65
Asym	17.16	10.43	17.84	16.92	22.74
MT-Tri	16.43	11.08	17.29	16.72	23.13
FLORS*	17.19	15.13	21.97	21.06	21.65

Table 5: Accuracy scores on dev sets for OOV and unknown word-tag (UWT) tokens.

the baseline significantly on only 2/5 domains (answers and emails). The FLORS tagger (Yin et al., 2015) fares better. Its contextual distributional features are particularly helpful on unknown word-tag combinations (see § 3.4), which is a limitation of the lexicalized generic bi-LSTM tagger.

For the high-data setup (Table 4) results are similar. Disagreement, however, is only favorable in the low-data setups; the effect of avoiding easy points no longer holds in the full data setup. Classic tri-training is the best method. In particular, traditional tri-training is complementary to word embedding initialization, pushing the non-pre-trained baseline to the level of SRC with Glove initalization. Tri-training pushes performance even further and results in the best model, significantly outperforming the baseline again in 4/5 cases, and reaching FLORS performance on weblogs. Multi-task tri-training is often slightly more effective than asymmetric tri-training (Saito et al., 2017); however, improvements for both are not robust across domains, sometimes performance even drops. The model likely is too simplistic for such a high-data POS setup, and exploring shared-private models might prove more fruitful (Liu et al., 2017). On the test sets, tri-training performs consistently the best.

POS analysis We analyze POS tagging accuracy with respect to word frequency[6] and unseen word-tag combinations (UWT) on the dev sets. Table 5 (top rows) provides percentage of unknown tags, OOVs and unknown word-tag (UWT) rate. The SANCL dataset is overall very challenging: OOV rates are high (6.8-11% compared to 2.3% in WSJ), so is the unknown word-tag (UWT) rate (answers and emails contain 2.91% and 3.47% UWT compared to 0.61% on WSJ) and almost all target domains even contain unknown tags (Schnabel and Schütze, 2014) (unknown tags: ADD,GW,NFP,XX), except for weblogs. Email is the domain with the highest OOV rate and highest unknown-tag-for-known-words rate. We plot accuracy with respect to word frequency on email in Figure 3, analyzing how the three methods fare in comparison to the baseline on this difficult domain.

[6]The binned log frequency was calculated with base 2 (bin 0 are OOVs, bin 1 are singletons and rare words etc).

Figure 3: POS accuracy per binned log frequency.

Regarding OOVs, the results in Table 5 (second part) show that classic tri-training outperforms the source model (trained on only source data) on 3/5 domains in terms of OOV accuracy, except on two domains with high OOV rate (newsgroups and weblogs). In general, we note that tri-training works best on OOVs and on low-frequency tokens, which is also shown in Figure 3 (leftmost bins). Both other methods fall typically below the baseline in terms of OOV accuracy, but MT-Tri still outperforms Asym in 4/5 cases. Table 5 (last part) also shows that no bootstrapping method works well on unknown word-tag combinations. UWT tokens are very difficult to predict correctly using an unsupervised approach; the less lexicalized and more context-driven approach taken by FLORS is clearly superior for these cases, resulting in higher UWT accuracies for 4/5 domains.

4 Related work

Learning under Domain Shift There is a large body of work on domain adaptation. Studies on unsupervised domain adaptation include early work on *bootstrapping* (Steedman et al., 2003; McClosky et al., 2006a), *shared feature representations* (Blitzer et al., 2006, 2007) and *instance weighting* (Jiang and Zhai, 2007). Recent ap-

proaches include *adversarial learning* (Ganin et al., 2016) and *fine-tuning* (Sennrich et al., 2016). There is almost no work on bootstrapping approaches for recent neural NLP, in particular under domain shift. Tri-training is less studied, and only recently re-emerged in the vision community (Saito et al., 2017), albeit is not compared to classic tri-training.

Neural network ensembling Related work on self-ensembling approaches includes snapshot ensembling (Huang et al., 2017) or temporal ensembling (Laine and Aila, 2017). In general, the line between "explicit" and "implicit" ensembling (Huang et al., 2017), like dropout (Srivastava et al., 2014) or temporal ensembling (Saito et al., 2017), is more fuzzy. As we noted earlier our multi-task learning setup can be seen as a form of self-ensembling.

Multi-task learning in NLP Neural networks are particularly well-suited for MTL allowing for parameter sharing (Caruana, 1993). Recent NLP conferences witnessed a "tsunami" of deep learning papers (Manning, 2015), followed by what we call a multi-task learning "wave": MTL has been successfully applied to a wide range of NLP tasks (Cohn and Specia, 2013; Cheng et al., 2015; Luong et al., 2015; Plank et al., 2016; Fang and Cohn, 2016; Søgaard and Goldberg, 2016; Ruder et al., 2017; Augenstein et al., 2018). Related to it is the pioneering work on adversarial learning (DANN) (Ganin et al., 2016). For sentiment analysis we found tri-training and our MT-Tri model to outperform DANN. Our MT-Tri model lends itself well to shared-private models such as those proposed recently (Liu et al., 2017; Kim et al., 2017), which extend upon (Ganin et al., 2016) by having separate source and target-specific encoders.

5 Conclusions

We re-evaluate a range of traditional general-purpose bootstrapping algorithms in the context of neural network approaches to semi-supervised learning under domain shift. For the two examined NLP tasks classic tri-training works the best and even outperforms a recent state-of-the-art method. The drawback of tri-training it its time and space complexity. We therefore propose a more efficient multi-task tri-training model, which outperforms both traditional tri-training and recent alternatives in the case of sentiment analysis. For POS tagging, classic tri-training is superior, performing especially well on OOVs and low frequency tokens, which suggests it is less affected by error propagation. Overall we emphasize the importance of comparing neural approaches to strong baselines and reporting results across several runs.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. Sebastian is supported by Irish Research Council Grant Number EBPPG/2014/30 and Science Foundation Ireland Grant Number SFI/12/RC/2289. Barbara is supported by NVIDIA corporation and thanks the Computing Center of the University of Groningen for HPC support.

References

Steven Abney. 2007. *Semisupervised learning for computational linguistics*. CRC Press.

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2018. Multi-task Learning of Pairwise Sequence Classification Tasks Over Disparate Label Spaces. In *Proceedings of NAACL-HLT 2018*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Annual Meeting-Association for Computational Linguistics*, 45(1):440.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP '06)*, pages 120–128.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain Separation Networks. *NIPS*.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. 2006. *Semi-Supervised Learning*, volume 1. MIT press.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-domain name error detection using a multi-task rnn. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 737–746. Association for Computational Linguistics.

Trevor Cohn and Lucia Specia. 2013. Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers), pages 32–42, Sofia, Bulgaria. Association for Computational Linguistics.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. *arXiv preprint arXiv:1706.09733*.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of ICLR 2017*.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Meng Fang and Trevor Cohn. 2016. Learning when to trust distant supervision: An application to low-resource pos tagging using cross-lingual projection. In *Proceedings of CoNLL-16*.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 587–593. Association for Computational Linguistics.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Francois Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17:1–35.

Rob van der Goot, Barbara Plank, and Malvina Nissim. 2017. To normalize, or not to normalize: The impact of normalization on part-of-speech tagging. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 31–39, Copenhagen, Denmark. Association for Computational Linguistics.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. *Proceedings of ICML 2017*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL 2018*.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot Ensembles: Train 1, get M for free. In *Proceedings of ICLR 2017*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Adversarial adaptation of synthetic or stale data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1297–1307, Vancouver, Canada. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Samuli Laine and Timo Aila. 2017. Temporal Ensembling for Semi-Supervised Learning. In *Proceedings of ICLR 2017*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *NAACL-HLT 2016*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Vancouver, Canada. Association for Computational Linguistics.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. 2015. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Christopher D Manning. 2015. Computational linguistics and deep learning. *Computational Linguistics*, 41(4):701–707.

David McClosky, Eugene Charniak, and Mark Johnson. 2006a. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, New York City, USA. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2006b. Reranking and Self-Training for Parser Adaptation. *International Conference on Computational Linguistics (COLING) and Annual Meeting of the Association for Computational Linguistics (ACL)*, (July):337–344.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the State of the Art of Evaluation in Neural Language Models. In *arXiv preprint arXiv:1707.05589*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, 59.

Barbara Plank. 2011. *Domain adaptation for parsing*. University Library Groningen.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 616–623.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*.

Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. 2017. Asymmetric Tri-training for Unsupervised Domain Adaptation. In *ICML 2017*.

Tobias Schnabel and Hinrich Schütze. 2014. FLORS: Fast and Simple Domain Adaptation for Part-of-Speech Tagging. *TACL*, 2:15–26.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Anders Søgaard. 2010. Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 205–208.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 231–235, Berlin, Germany. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Mark Steedman, Rebecca Hwa, Stephen Clark, Miles Osborne, Anoop Sarkar, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Example selection for bootstrapping statistical parsers. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using gigaword scale unlabeled data. pages 665–673.

Vincent Van Asch and Walter Daelemans. 2016. Predicting the effectiveness of self-training: Application to sentiment classification. *arXiv preprint arXiv:1601.03288*.

Fangzhao Wu and Yongfeng Huang. 2016. Sentiment Domain Adaptation with Multiple Sources. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 301–310.

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*.

Wenpeng Yin, Tobias Schnabel, and Hinrich Schütze. 2015. Online Updating of Word Representations for Part-of-Speech Tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, September, pages 1329–1334.

Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang, and Tingting He. 2016. Bi-transferring deep neural networks for domain adaptation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 322–332, Berlin, Germany. Association for Computational Linguistics.

Zhi-Hua Zhou and Ming Li. 2005. Tri-Training: Exploiting Unlabeled Data Using Three Classifiers. *IEEE Trans.Data Eng.*, 17(11):1529–1541.

Xiaojin Zhu. 2005. Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

Xiaojin Zhu and Andrew B Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.

SemEval-2020 Task 11: Detection of Propaganda Techniques in News Articles

Giovanni Da San Martino[1], **Alberto Barrón-Cedeño**[2],
Henning Wachsmuth[3], **Rostislav Petrov**[4] and **Preslav Nakov**[1]
[1]Qatar Computing Research Institute, HBKU, Qatar [2]Università di Bologna, Forlì, Italy
[3]Paderborn University, Paderborn, Germany [4]A Data Pro, Sofia, Bulgaria
{gmartino, pnakov}@hbku.edu.qa a.barron@unibo.it
henningw@upb.de rostislav.petrov@adata.pro

Abstract

We present the results and the main findings of SemEval-2020 Task 11 on Detection of Propaganda Techniques in News Articles. The task featured two subtasks. Subtask SI is about *Span Identification*: given a plain-text document, spot the specific text fragments containing propaganda. Subtask TC is about *Technique Classification*: given a specific text fragment, in the context of a full document, determine the propaganda technique it uses, choosing from an inventory of 14 possible propaganda techniques. The task attracted a large number of participants: **250** teams signed up to participate and **44** made a submission on the test set. In this paper, we present the task, analyze the results, and discuss the system submissions and the methods they used. For both subtasks, the best systems used pre-trained Transformers and ensembles.

1 Introduction

Propaganda aims at influencing people's mindset with the purpose of advancing a specific agenda. It can hide in news published by both established and non-established outlets, and, in the Internet era, it has the potential of reaching very large audiences (Muller, 2018; Tardáguila et al., 2018; Glowacki et al., 2018). Propaganda is most successful when it goes unnoticed by the reader, and it often takes some training for people to be able to spot it. The task is way more difficult for inexperienced users, and the volume of text produced on a daily basis makes it difficult for experts to cope with it manually. With the recent interest in "fake news", the detection of propaganda or highly biased texts has emerged as an active research area. However, most previous work has performed analysis at the document level only (Rashkin et al., 2017; Barrón-Cedeño et al., 2019a) or has analyzed the general patterns of online propaganda (Garimella et al., 2015; Chatfield et al., 2015).

SemEval-2020 Task 11 offers a different perspective: a fine-grained analysis of the text that complements existing approaches and can, in principle, be combined with them. Propaganda in text (and in other channels) is conveyed through the use of diverse propaganda techniques (Miller, 1939), which range from leveraging on the emotions of the audience —such as using *loaded language* or *appeals to fear*— to using logical fallacies —such as *straw men* (misrepresenting someone's opinion), hidden *ad-hominem fallacies*, and *red herring* (presenting irrelevant data). Some of these techniques have been studied in tasks such as hate speech detection (Gao et al., 2017) and computational argumentation (Habernal et al., 2018).

Figure 1 shows the fine-grained propaganda identification pipeline, including the two targeted subtasks. Our goal is to facilitate the development of models capable of spotting text fragments where propaganda techniques are used. The task featured the following subtasks:

Subtask SI *(Span Identification)*: Given a plain-text document, identify those specific fragments that contain at least one propaganda technique. (This is a binary sequence tagging task.)

Subtask TC *(Technique Classification)*: Given a propagandistic text snippet and its document context, identify the propaganda technique used in that snippet. (This is a multi-class classification problem.)

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 1377–1414
Barcelona, Spain (Online), December 12, 2020.

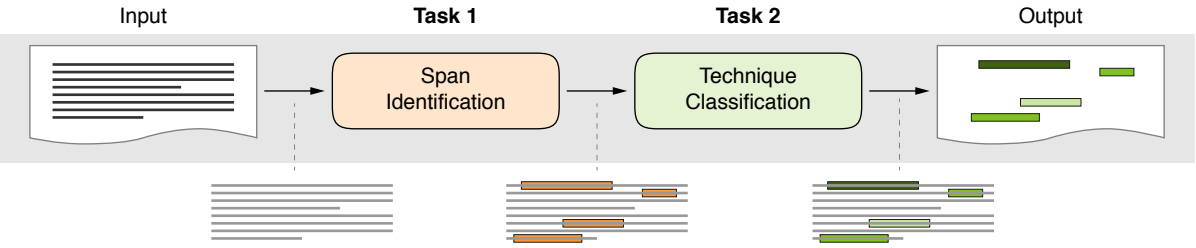


Figure 1: The full propaganda identification pipeline, including the two subtasks: Span Identification and Technique Classification.

A total of 250 teams registered for the task, 44 of them made an official submission on the test set (66 submissions for both subtasks), and 32 of the participating teams submitted a system description paper.

The rest of the paper is organized as follows. Section 2 introduces the propaganda techniques we considered in this shared task. Section 3 describes the organization of the task, the corpus and the evaluation measures. An overview of the participating systems is given in Section 4, while Section 5 discusses the evaluation results. Related work is described in Section 6. Finally, Section 7 draws some conclusions, and discusses some directions for future work.

2 Propaganda and its Techniques

Propaganda comes in many forms, but it can be recognized by its persuasive function, sizable target audience, the representation of a specific group's agenda, and the use of faulty reasoning and/or emotional appeals (Miller, 1939). The term *propaganda* was coined in the 17th century, and initially referred to the propagation of the Catholic faith in the New World (Jowett and O'Donnell, 2012a, p. 2). It soon took a pejorative connotation, as its meaning was extended to also mean opposition to Protestantism. In more recent times, the Institute for Propaganda Analysis (Ins, 1938) proposed the following definition:

> **Propaganda.** *Expression of opinion or action by individuals or groups deliberately designed to influence opinions or actions of other individuals or groups with reference to predetermined ends.*

Recently, Bolsover and Howard (2017) dug deeper into this definition identifying its two key elements: (*i*) trying to influence opinion, and (*ii*) doing so on purpose.

Propaganda is a broad concept, which runs short for the aim of annotating specific propaganda fragments. Yet, influencing opinions is achieved through a series of rhetorical and psychological techniques, and in the present task, we focus on identifying the use of such techniques in text. Whereas the definition of propaganda is widely accepted in the literature, the set of propaganda techniques considered, and to some extent their definition, differ between different scholars (Torok, 2015). For instance, Miller (1939) considers seven propaganda techniques, whereas Weston (2000) lists at least 24 techniques, and the Wikipedia article on the topic includes 67.[1] Below, we describe the propaganda techniques we consider in the task: a curated list of fourteen techniques derived from the aforementioned studies. We only include techniques that can be found in journalistic articles and can be judged intrinsically, without the need to retrieve supporting information from external resources. For example, we do not include techniques such as *card stacking* (Jowett and O'Donnell, 2012b, p. 237), since it would require comparing multiple sources. Note that our list of techniques was initially longer than fourteen, but we decided, after the annotation phase, to merge similar techniques with very low frequency in the corpus. A more detailed list with definitions and examples is available online[2] and in Appendix C, and examples are shown in Table 1.

1. Loaded language. Using specific words and phrases with strong emotional implications (either positive or negative) to influence an audience (Weston, 2000, p. 6).

2. Name calling or labeling. Labeling the object of the propaganda campaign as either something the target audience fears, hates, finds undesirable or loves, praises (Miller, 1939).

[1] `https://en.wikipedia.org/wiki/Propaganda_techniques`; last visit February 2019.
[2] `http://propaganda.qcri.org/annotations/definitions.html`

#	Technique	Snippet
1	Loaded language	**Outrage** as Donald Trump suggests injecting disinfectant to kill virus.
2	Name calling, labeling	WHO: Coronavirus emergency is '**Public Enemy Number 1**'
3	Repetition	I still have a **dream**. It is a **dream** deeply rooted in the American **dream**. I have a **dream** that one day . . .
4	Exaggeration, minimization	Coronavirus **'risk to the American people remains very low'**, Trump said.
5	Doubt	**Can the same be said for the Obama Administration**?
6	Appeal to fear/prejudice	**A dark, impenetrable and "irreversible" winter of persecution of the faithful by their own shepherds will fall**.
7	Flag-waving	Mueller attempts **to stop the will of We the People**!!! It's time to jail Mueller.
8	Causal oversimplification	**If France had not have declared war on Germany then World War II would have never happened.**
9	Slogans	**"BUILD THE WALL!"** Trump tweeted.
10	Appeal to authority	**Monsignor Jean-Franois Lantheaume, who served as first Counsellor of the Nunciature in Washington, confirmed that "Vigan said the truth. That's all."**
11	Black-and-white fallacy	Francis said these words: "**Everyone is guilty for the good he could have done and did not do . . . If we do not oppose evil, we tacitly feed it**."
12	Thought-terminating cliché	**I do not really see any problems there.** Marx is the President.
13	Whataboutism	President Trump —**who himself avoided national military service** in the 1960's— keeps beating the war drums over North Korea.
	Straw man	"Take it seriously, but with a large grain of salt." **Which is just Allen's more nuanced way of saying: "Don't believe it**."
	Red herring	**"You may claim that the death penalty is an ineffective deterrent against crime – but what about the victims of crime? How do you think surviving family members feel when they see the man who murdered their son kept in prison at their expense? Is it right that they should pay for their son's murderer to be fed and housed?"**
14	Bandwagon	He tweeted, "**EU no longer considers #Hamas a terrorist group. Time for US to do same."**
	Reductio ad hitlerum	"Vichy journalism," a term which now fits so much of the mainstream media. **It collaborates in the same way that the Vichy government in France collaborated with the Nazis.**

Table 1: The 14 propaganda techniques with examples, where the propaganda span is shown in bold.

3. Repetition. Repeating the same message over and over again, so that the audience will eventually accept it (Torok, 2015; Miller, 1939).

4. Exaggeration or minimization. Either representing something in an excessive manner: making things larger, better, worse or making something seem less important or smaller than it actually is (Jowett and O'Donnell, 2012b, pag. 303).

5. Doubt. Questioning the credibility of someone or something.

6. Appeal to fear/prejudice. Seeking to build support for an idea by instilling anxiety and/or panic in the population towards an alternative, possibly based on preconceived judgments.

7. Flag-waving. Playing on strong national feeling (or with respect to any group, e.g., race, gender, political preference) to justify or promote an action or idea (Hobbs and Mcgee, 2008).

8. Causal oversimplification. Assuming a single cause or reason when there are multiple causes behind an issue. We include in the definition also *scapegoating*, e.g., transferring the blame to one person or group of people without investigating the complexities of an issue.

9. Slogans. A brief and striking phrase that may include labeling and stereotyping. Slogans tend to act as emotional appeals (Dan, 2015).

10. Appeal to authority. Stating that a claim is true simply because a valid authority or expert on the issue supports it, without any other supporting evidence (Goodwin, 2011). We include in this technique the special case in which the reference is not an authority or an expert, although it is referred to as *testimonial* in the literature (Jowett and O'Donnell, 2012b, pag. 237).

11. Black-and-white fallacy, dictatorship. Presenting two alternative options as the only possibilities, when in fact more possibilities exist (Torok, 2015). *Dictatorship* is an extreme case: telling the audience exactly what actions to take, eliminating any other possible choice.

12. Thought-terminating cliché. Words or phrases that discourage critical thought and meaningful discussion on a topic. They are typically short, generic sentences that offer seemingly simple answers to complex questions or that distract attention away from other lines of thought (Hunter, 2015, p. 78).

13. Whataboutism, straw man, red herring. Here we merge together three techniques, which are relatively rare taken individually: (*i*) *Whataboutism:* Discredit an opponent's position by charging them with hypocrisy without directly disproving their argument (Richter, 2017). (*ii*) *Straw man:* When an opponent's proposition is substituted with a similar one, which is then refuted instead of the original (Walton, 2013). Weston (2000, p. 78) specifies the characteristics of the substituted proposition: "caricaturing an opposing view so that it is easy to refute". (*iii*) *Red herring:* Introducing irrelevant material to the issue being discussed, so that everyone's attention is diverted away from the points made (Weston, 2000, p. 78).

14. Bandwagon, reductio ad hitlerum. Here we merge together two techniques, which are relatively rare taken individually: (*i*) *Bandwagon.* Attempting to persuade the target audience to join in and take the course of action because "everyone else is taking the same action" (Hobbs and Mcgee, 2008). (*ii*) *Reductio ad hitlerum:* Persuading an audience to disapprove an action or idea by suggesting that it is popular with groups hated in contempt by the target audience. It can refer to any person or concept with a negative connotation (Teninbaum, 2009).

We provided the definitions, together with some examples and an annotation schema, to professional annotators, and we asked them to manually annotate selected news articles. The annotators worked with an earlier version of the annotation schema, which contained eighteen techniques (Da San Martino et al., 2019b). As some of these techniques were quite rare, which could cause data sparseness issues for the participating systems, for the purpose of the present SemEval-2020 task 11, we decided to get rid of the four rarest techniques. In particular, we merged *Red herring* and *Straw man* with *Whataboutism* (under technique 13), since all three techniques are trying to divert the attention to an irrelevant topic and away from the actual argument. We further merged *Bandwagon* with *Reductio ad hitlerum* (under technique 14), since they both try to approve/disapprove an action or idea by pointing to what is popular/unpopular. Finally, we dropped one rare technique, which we could not easily merge with other techniques: *Obfuscation, Intentional vagueness, Confusion*. As a result, we reduced the eighteen original propaganda techniques to fourteen.

3 Evaluation Framework

The SemEval 2020 Task 11 evaluation framework consists of the PTC-SemEval20 corpus and the evaluation measures for both the span identification and the technique classification subtasks. We describe the organization of the task in Section 3.3; here, we focus on the dataset, the evaluation measure, and the organization setup.

3.1 The PTC-SemEval20 Corpus

In order to build the PTC-SemEval20 corpus, we retrieved a sample of news articles from the period starting in mid-2017 and ending in early 2019. We selected 13 propaganda and 36 non-propaganda news media outlets, as labeled by Media Bias/Fact Check,[3] and we retrieved articles from these sources. We deduplicated the articles on the basis of word n-gram matching (Barrón-Cedeño and Rosso, 2009), and we discarded faulty entries, e.g., empty entries from blocking websites.

The annotation job consisted of both spotting a propaganda snippet and, at the same time, labeling it with a specific propaganda technique. The annotation guidelines are shown in Appendix C; they are also available online.[4] We ran the annotation in two phases: (*i*) two annotators labeled an article independently, and (*ii*) the same two annotators gathered together with a *consolidator* to discuss dubious instances, e.g., spotted only by one annotator, boundary discrepancies, label mismatch, etc. This protocol was designed after a pilot annotation stage, in which a relatively large number of snippets had been spotted by one annotator only.

[3] An initiative where professional journalists profile news outlets; `https://mediabiasfactcheck.com`.

[4] `https://propaganda.qcri.org/annotations/`

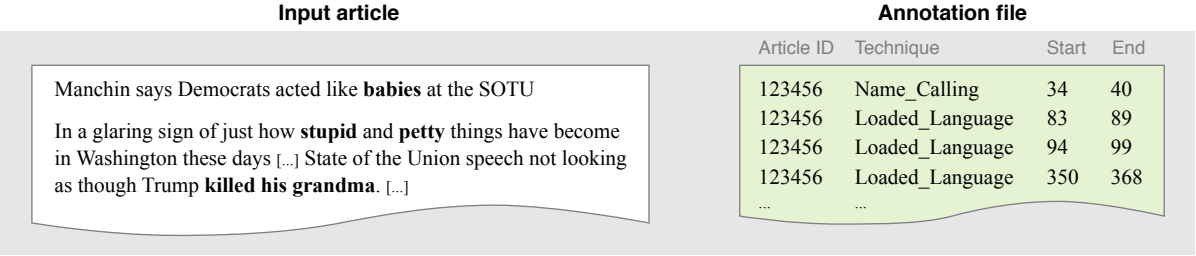


Figure 2: Example of a plain-text article (left) and its annotation (right). The *Start* and the *End* columns are the indices representing the character span of the spotted technique.

partition	articles	average lengths		propaganda
		chars	tokens	snippets
training	371	5,681±5,425	927±899	6,128
development	75	4,700±2,904	770±473	1,063
test	90	4,518±2,602	744±433	1,790
all	536	5,348±4,789	875±793	8,981

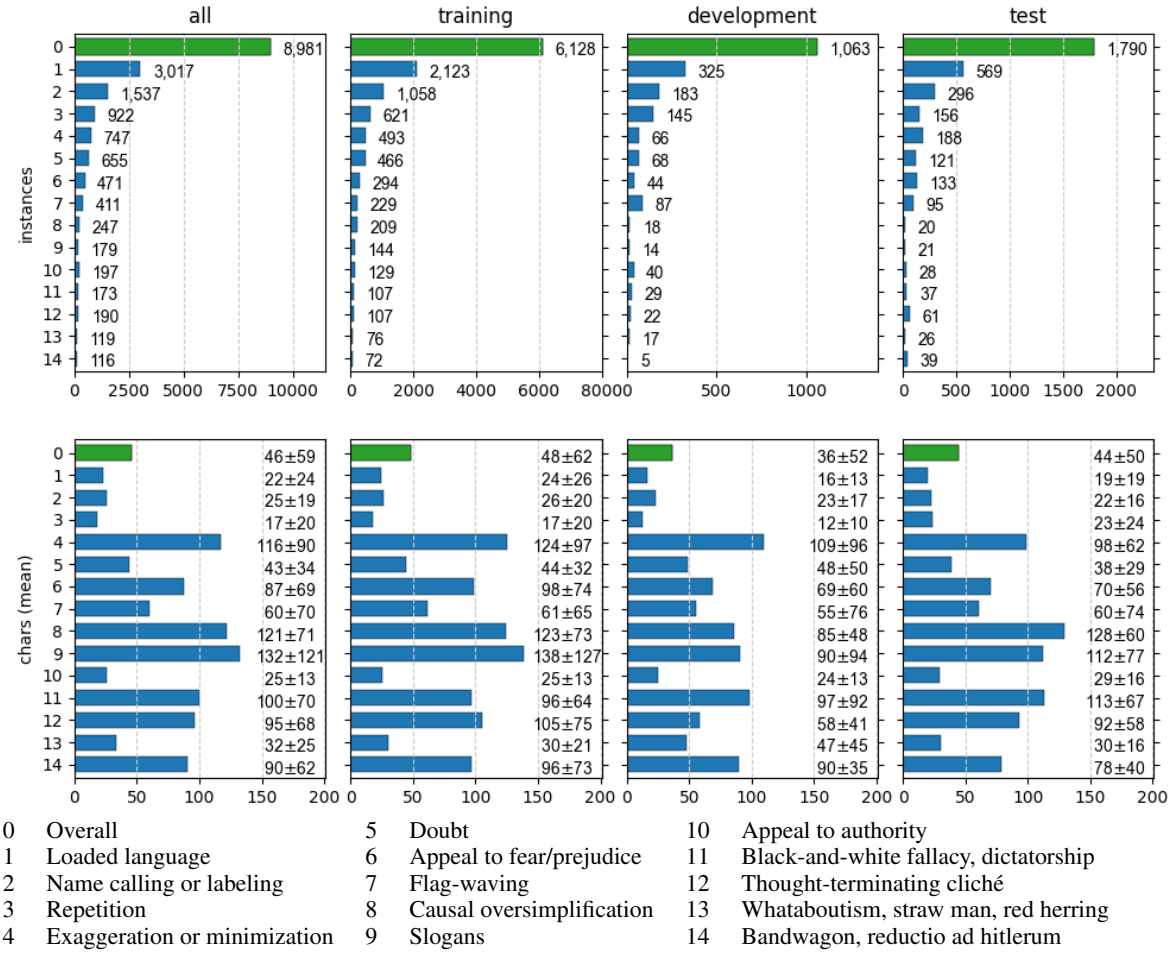
Table 2: Statistics about the train/dev/test parts of the PTC-SemEval20 corpus, including the number of articles, their average lengths in terms of characters and tokens, and the total number of propaganda snippets they contain.

The annotation team consisted of six professional annotators from A Data Pro,[5] trained to spot and to label the propaganda snippets in free text. The job was carried out on an instance of the Anafora annotation platform (Chen and Styler, 2013), which we tailored for our propaganda annotation task. Figure 2 shows an example of an article and its annotations.

We evaluated the quality of the annotation process in terms of γ agreement (Mathet et al., 2015) between each of the annotators and the final gold labels. The γ agreement on the annotated articles is on average 0.6; see (Da San Martino et al., 2019b) for a more detailed discussion of inter-annotator agreement. The training and the development part of the PTC-SemEval20 corpus are the same as the training and the testing datasets described in (Da San Martino et al., 2019b). The test part of the PTC-SemEval20 corpus consists of 90 additional articles selected from the same sources as for training and development. For the test articles, we further extended the annotation process by adding one extra consolidation step: we revisited all the articles in that partition and we performed the necessary adjustments to the spans and to the labels as necessary, after a thorough discussion and convergence among at least three experts who were not involved in the initial annotations.

Table 2 shows some statistics about the corpus we use for the task. It is worth noting that a number of propaganda snippets of different classes overlap. Hence, the number of snippets for the span identification subtask is smaller (e.g., 1,405 for the span identification subtask vs. 1,790 for the technique classification subtask on the test set). The full collection of 536 articles contains 8,981 propaganda text snippets, belonging to one of the above-described fourteen classes. Figure 3 zooms into such snippets and shows the number of instances and the mean length for each class. We can see that, by a large margin, the most common propaganda technique in our news articles is *Loaded Language*, which is about twice as frequent as the second most frequent technique: *Name Calling or Labeling*. Whereas these two techniques are among the ones that are expressed in the shortest spans, other propaganda techniques such as *Exaggeration*, *Causal Oversimplification*, and *Slogans* tend to be the longest.

[5] https://www.aiidatapro.com



0	Overall	5	Doubt	10	Appeal to authority
1	Loaded language	6	Appeal to fear/prejudice	11	Black-and-white fallacy, dictatorship
2	Name calling or labeling	7	Flag-waving	12	Thought-terminating cliché
3	Repetition	8	Causal oversimplification	13	Whataboutism, straw man, red herring
4	Exaggeration or minimization	9	Slogans	14	Bandwagon, reductio ad hitlerum

Figure 3: Statistics about the propaganda snippets in the different partitions of the PTC-SemEval20 corpus. Top: number of instances per class. Bottom: mean snippet length per class.

3.2 Evaluation Measures

Subtask SI Evaluating subtask SI requires us to match text spans. Our SI evaluation function gives credit to partial matches between gold and predicted spans.

Let d be a news article in a set D. A gold span t is a sequence of contiguous indices of the characters composing a text fragment $t \subseteq d$. For example, in Figure 4 (top-left) the gold fragment *"stupid and petty"* is represented by the set of indices $t_1 = [4, 19]$. We denote with $T_d = \{t_1, \ldots, t_n\}$ the set of all gold spans for an article d and with $T = \{T_d\}_d$ the set of all gold annotated spans in D. Similarly, we define $S_d = \{s_1, \ldots, s_m\}$ and S to be the set of predicted spans for an article d and a dataset D, respectively. We compute precision P and recall R by adapting the formulas in (Potthast et al., 2010):

$$P(S,T) = \frac{1}{|S|} \cdot \sum_{d \in D} \sum_{s \in S_d, t \in T_d} \frac{|(s \cap t)|}{|t|}, \quad (1)$$

$$R(S,T) = \frac{1}{|T|} \cdot \sum_{d \in D} \sum_{s \in S_d, t \in T_d} \frac{|(s \cap t)|}{|s|}. \quad (2)$$

We define Eq. (1) to be zero when $|S| = 0$ and Eq. (2) to be zero when $|T| = 0$. Notice that the predicted spans may overlap, e.g., spans s_3 and s_4 in Figure 4. Therefore, in order for Eq. 1 and Eq. 2 to get values lower than or equal to 1, all overlapping annotations, independently of their techniques, are merged first. For example, s_3 and s_4 are merged into one single annotation, corresponding to s_4.

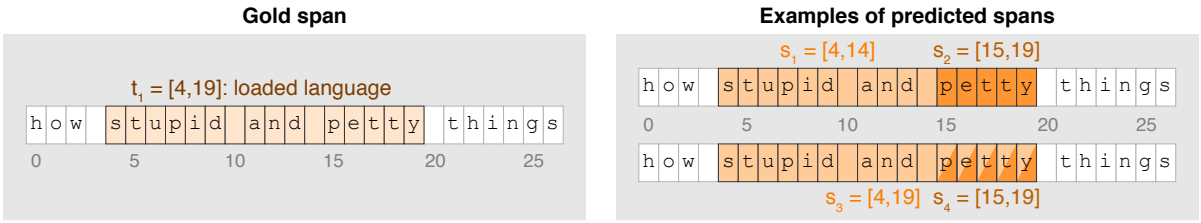


Figure 4: Example of equivalent annotations for the Span Identification subtask.

Finally, the evaluation measure for subtask SI is the F_1 score, defined as the harmonic mean between $P(S,T)$ and $R(S,T)$:

$$F_1(S,T) = 2 \cdot \frac{P(S,T) \cdot R(S,T)}{P(S,T) + R(S,T)}. \quad (3)$$

Subtask TC Given a propaganda snippet in an article, subtask TC asks to identify the technique in it. Since there are identical spans annotated with different techniques (around 1.8% of the total annotations), formally this is a multi-label multi-class classification problem. However, we decided to consider the problem as a single-label multi-class one, by performing the following adjustments: (*i*) whenever a span is associated with multiple techniques, the input file will have multiple copies of such fragments and (*ii*) the evaluation function ensures that the best match between the predictions and the gold labels for identical spans is used for the evaluation. In other words, the evaluation score is not affected by the order in which the predictions for identical spans are submitted.

The evaluation measure for subtask TC is micro-average F_1. Note that as we have converted this into a single-label task, micro-average F_1 is equivalent to Accuracy (as well as to Precision and to Recall).

3.3 Task Organization

We ran the shared task in two phases:

Phase 1. Only training and development data were made available, and no gold labels were provided for the latter. The participants competed against each other to achieve the best performance on the development set. A live leaderboard was made available to keep track of all submissions.

Phase 2. The test set was released and the participants were given just a few days to submit their final predictions. The release of the test set was done task-by-task, since giving access to the input files for the TC subtask would have disclosed the gold spans for the SI subtask.

In phase 1, the participants could make an unlimited number of submissions on the development set, and they could see the outcomes in their private space. The best team score, regardless of the submission time, was also shown in a public leaderboard. As a result, not only could the participants observe the impact of various modifications in their own systems, but they could also compare against the results by other participating teams. In phase 2, the participants could again submit multiple runs, but they did not get any feedback on their performance. Only the last submission of each team was considered official and was used for the final team ranking.

In phase 1, a total of 47 teams made submissions on the development set for the SI subtask, and 46 teams submitted for the TC subtask. In phase 2, the number of teams who made official submissions on the test set for subtasks SI and TC was 35 and 31, respectively: this is a total of 66 submissions for the two subtasks, which were made by 44 different teams.

Note that we left the submission system open for submissions on the development set (phase 1) after the competition was over. The up-to-date leaderboards can be found on the website of the competition.[6]

[6] http://propaganda.qcri.org/semeval2020-task11/leaderboard.php

4 Participating Systems

In this section, we focus on a general description of the systems participating on both the SI and the TC subtasks. We pay special attention to the most successful approaches. The subindex on the right of each team represents their official rank in the subtasks. Appendix A includes brief descriptions of all systems.

4.1 Span Identification Subtask

Table 3 shows a quick overview of the systems that took part in the SI subtask.[7] All systems in the top-10 positions relied on some kind of Transformer, in combination with an LSTM or a CRF. In most cases, the Transformer-generated representations were complemented by engineered features, such as named entities and the presence of sentiment and subjectivity clues.

Team **Hitachi** (SI:1) achieved the top performance in this subtask (Morio et al., 2020). They used a BIO encoding, which is typical for related segmentation and labeling tasks (e.g., named entity recognition). They relied upon a complex heterogeneous multi-layer neural network, trained end-to-end. The network uses pre-trained language models, which generate a representation for each input token. They further added part-of-speech (PoS) and named entity (NE) embeddings. As a result, there are three representations for each token, which are concatenated and used as an input to bi-LSTMs. At this moment, the network branches, as it is trained with three objectives: (*i*) the main BIO tag prediction objective and two auxiliary ones, namely (*ii*) token-level technique classification, and (*iii*) sentence-level classification. There is one Bi-LSTM for objectives (*i*) and (*ii*), and there is another Bi-LSTM for objective (*iii*). For the former, they used an additional CRF layer, which helps improve the consistency of the output. A number of architectures were trained independently —using BERT, GPT-2, XLNet, XLM, RoBERTa, or XLM-RoBERTa—, and the resulting models were combined in ensembles.

Team **ApplicaAI** (SI:2) (Jurkiewicz et al., 2020) based its success on self-supervision using the RoBERTa model. They used a RoBERTa-CRF architecture trained on the provided data and used it to iteratively produce silver data by predicting on 500k sentences and retraining the model with both gold and silver data. The final classifier was an ensemble of models trained on the original corpus, re-weighting, and a model trained also on silver data. ApplicaAI was not the only team that reported performance boost when using additional data. Team **UPB** (SI:5) (Paraschiv and Cercel, 2020) decided not to stick to the pre-trained models from BERT–base alone and used masked language modeling to domain-adapt it using 9M articles containing fake, suspicious, and hyperpartisan news articles. Team **DoNotDistribute** (SI:22) (Kranzlein et al., 2020) also opted for generating silver data, but with a different strategy. They report a 5% performance boost when adding 3k new silver training instances. To produce them, they used a library to create near-paraphrases of the propaganda snippets by randomly substituting certain PoS words. Team **SkoltechNLP** (SI:25) (Dementieva et al., 2020) performed data augmentation based on distributional semantics. Finally, team **WMD** (SI:33) (Daval-Frerot and Yannick, 2020) applied multiple strategies to augment the data such as back translation, synonym replacement and TF.IDF replacement (replace unimportant words, based on TF.IDF score, by other unimportant words).

Closing the top-three submissions, Team **aschern** (SI:3) (Chernyavskiy et al., 2020) fine-tuned an ensemble of two differently intialized RoBERTa models, each with an attached CRF for sequence labeling and span character boundary post-processing.

There have been several other promising strategies. Team **LTIatCMU** (SI:4) (Khosla et al., 2020) used a multi-granular BERT BiLSTM model with additional syntactic and semantic features at the word, sentence and document level, including PoS, named entities, sentiment, and subjectivity. It was trained jointly for token and sentence propaganda classification, with class balancing. They further fine-tuned BERT on persuasive language using 10,000 articles from propaganda websites, which turned out to be important. Team **PsuedoProp** (SI:14) (Chauhan and Diddee, 2020) built a preliminary sentence-level classifier using an ensemble of XLNet and RoBERTa, before it fine-tuned a BERT-based CRF sequence tagger to identify the exact spans. Team **BPGC** (SI:21) (Patil et al., 2020) went beyond these multigranularity approaches. Information both at the article and at the sentence level was considered when classifying each word as propaganda or not, by computing and concatenating vectorial representations for the three inputs.

[7] Tables 3 and 4 only include the systems for which a description paper was submitted.

Rank. Team	Transformers									Learning Models								Representations											Misc		
	BERT	RoBERTa	XLNet	XLM	XLM RoBERTa	ALBERT	GPT-2	SpanBERT	LaserTagger	LSTM	CNN	SVM	Naïve Bayes	Boosting	Log regressor	Random forest	CRF	Embeddings	ELMo	NEs	Words/n-grams	Chars/n-grams	PoS	Trees	Sentiment	Subjectivity	Rhetorics	LIWC	Ensemble	Data augmentation	Post-processing
1. Hitachi	☑	☑	☑	☑	☑		☑				☑						☑	☑		☑			☑						☑		
2. ApplicaAI		☑															☑													☑	
3. aschern		☑															☑	☑				☑							☑		☑
4. LTIatCMU	☑									☑							☑	☑		☑	☑		☑	☑	☑	☑		☑			
5. UPB	☑									✔							☑	✔												☑	
7. NoPropaganda	☑								☑	✔							☑	☑	☑			☑									
8. CyberWallE	☑									☑		✔											☑		☑		☑				
9. Transformers	☑	☑								☑	☑								☑										☑		
11. YNUtaoxin	☑	✔	✔																												
13. newsSweeper	☑	✔					✔	✔															✔								
14. PsuedoProp	☑	☑	☑			✔																							☑		
16. YNUHPCC	✔									☑								☑													
17. NLFIIT	✔									☑							✔	☑											✔		
20. TTUI	☑	☑																		☑									☑		☑
21. BPGC	✔	☑	✔				✔											☑													
22. DoNotDistribute	☑									✔										☑	☑		☑							☑	
23. UTMN															☑			☑							☑						
25. syrapropa								☑										☑			☑										
26. SkoltechNLP	☑									✔	✔						✔		☑			☑									
27. NTUAAILS										☑									☑												
28. UAIC1860												✔	✔		✔	☑	☑	☑			☑	☑									
31. 3218IR											☑							☑													
33. WMD	☑											✔		✔		✔		☑											☑	☑	☑
– UNTLing																	✔	✔			✔		✔	✔	✔						

1. (Morio et al., 2020)
2. (Jurkiewicz et al., 2020)
3. (Chernyavskiy et al., 2020)
4. (Khosla et al., 2020)
5. (Paraschiv and Cercel, 2020)
7. (Dimov et al., 2020)
8. (Blaschke et al., 2020)
9. (Verma et al., 2020)
11. (Tao and Zhou, 2020)
13. (Singh et al., 2020)
14. (Chauhan and Diddee, 2020)
16. (Dao et al., 2020)
17. (Martinkovic et al., 2020)
20. (Kim and Bethard, 2020)
21. (Patil et al., 2020)
22. (Kranzlein et al., 2020)
23. (Mikhalkova et al., 2020)
25. (Li and Xiao, 2020)
26. (Dementieva et al., 2020)
27. (Arsenos and Siolas, 2020)
28. (Ermurachi and Gifu, 2020)
31. (Dewantara et al., 2020)
33. (Daval-Frerot and Yannick, 2020)
– (Krishnamurthy et al., 2020)

Table 3: Overview of the approaches to the span identification subtask. ☑=part of the official submission; ✔=considered in internal experiments. The references to the description papers appear at the bottom.

A large number of the participating teams built systems that rely heavily on engineered features. For instance, Team **CyberWallE**(SI:8) (Blaschke et al., 2020) used features modeling sentiment, rhetorical structure, and POS tags, while team **UTMN**(SI:23) injected the sentiment intensity from VADER and it was among the only teams not relying on deep learning architectures to produce a computationally affordable model.

4.2 Technique Classification Subtask

The same trends as for the snippet identification subtask can be observed in the approaches used for the technique classification subtask: practically, all top-performing approaches used representations produced by some kind of Transformer.

Team **ApplicaAI**(TC:1) achieved the top performance for this subtask (Jurkiewicz et al., 2020). As in their approach to subtask SI, ApplicaAI produced additional silver data for training. This time, they ran their high-performing SI model to spot new propaganda snippets in free text and applied their preliminary TC model to produce extra silver-labeled instances. Their final classifier consisted of an ensemble of models trained on the original corpus, re-weighting, and a model trained also on silver data. In all cases, the input to the classifiers consisted of propaganda snippets and their context.

Rank. Team	Transformers										Learning Models										Representations											Misc		
	BERT	R BERT	RoBERTa	XLNet	XLM	XLM RoBERTa	ALBERT	GPT-2	SpanBERT	DistilBERT	LSTM	RNN	CNN	SVM	Naïve Bayes	Boosting	Log regressor	Random forest	Regression tree	CRF	XGBoost	Embeddings	ELMo	NEs	Words/n-grams	Chars/n-grams	PoS	Sentiment	Rhetorics	Lexicons	String matching	Topics	Ensemble	Data augmentation	Post-processing
1. ApplicaAI			☑																	☑												☑	☑	
2. aschern			☑																			☑										☑	☑	☑
3. Hitachi	☑		☑	☑	☑	☑	☑						☑									☑		☑			☑					☑	☑	
4. Solomon			☑											✔								☑									☑		☑	
5. newsSweeper	✔		☑					✔	✔																	✔								
6. NoPropaganda	☑	☑																																
7. Inno	✔		☑	✔						☑	✔						✔						✔											
8. CyberWallE	☑																							☑				☑						☑
10. Duth	☑																							☑										
11. DiSaster	☑											✔					☑								☑								☑	
13. SocCogCom	☑						✔												✔								☑		✔					
14. TTUI	☑		☑																						☑							☑		
15. JUST	☑																					☑												
16. NLFIIT	✔										☑									✔		✔	☑										✔	
17. UMSIForeseer	☑																					☑											☑	
18. BPGC	☑										✔		✔				☑					✔			☑	☑						☑	☑	
19. UPB	☑																																☑	
20. syrapropa	☑																☑								☑		☑							☑
21. WMD	☑										✔			✔		✔		✔				☑											☑	☑	
22. YNUHPCC	☑										☑										✔														
24. DoNotDistribute	☑										☑													☑	☑		☑							
25. NTUAAILS											☑											☑												
26. UAIC1860														✔	✔		✔	☑				☑			☑	☑								
27. UNTLing																	☑					☑		☑	☑				☑					

1. (Jurkiewicz et al., 2020)
2. (Chernyavskiy et al., 2020)
3. (Morio et al., 2020)
4. (Raj et al., 2020)
5. (Singh et al., 2020)
6. (Dimov et al., 2020)
7. (Grigorev and Ivanov, 2020)
8. (Blaschke et al., 2020)
10. (Bairaktaris et al., 2020)
11. (Kaas et al., 2020)
13. (Krishnamurthy et al., 2020)
14. (Kim and Bethard, 2020)
15. (Altiti et al., 2020)
16. (Martinkovic et al., 2020)
17. (Jiang et al., 2020)
18. (Patil et al., 2020)
19. (Paraschiv and Cercel, 2020)
20. (Li and Xiao, 2020)
21. (Daval-Frerot and Yannick, 2020)
22. (Dao et al., 2020)
24. (Kranzlein et al., 2020)
25. (Arsenos and Siolas, 2020)
26. (Ermurachi and Gifu, 2020)
27. (Petee and Palmer, 2020)
29. (Verma et al., 2020)

Table 4: Overview of the approaches to the technique classification subtask. ☑=part of the official submission; ✔=considered in internal experiments. The references to the description papers appear at the bottom.

Team **aschern** (TC:2) (Chernyavskiy et al., 2020) was the second best, and it based its success on a RoBERTa ensemble with several interesting techniques. They treated the task as one of sequence classification, using an average embedding of the surrounding tokens and the length of the span as contextual features. They further incorporated knowledge from the span identification task, using transfer learning: namely, they first pre-trained their model for the SI task, and then continued training for the TC task. Finally, they performed task-specific postprocessing in order to increase the consistency for the repetition technique spans and to avoid insertions of techniques inside other techniques.

Team **Hitachi** (TC:3) (Morio et al., 2020) used two distinct feed-forward neural networks (FFNs). The first one is for sentence representation, whereas the second one is for the representation of the tokens in the propaganda span. The propaganda span representation is obtained by concatenating the representation of the begin-of-sentence token, the span start token, the span end token, and the aggregated representation obtained using attention and max-pooling. As for their winning approach to SI, team Hitachi trained on the TC subtask independently with different large-scale pre-trained state-of-the-art language models (BERT, GPT-2, XLNet, XLM, RoBERTa, or XLM-RoBERTa), and then combined the resulting models in an ensemble.

As the top-performing models to subtask TC show, while the two subtasks can be seen as fairly independent, combining them in a reasonable way pays back. Additionally, the context of a propaganda snippet is important to identify the specific propaganda technique it uses. Indeed, other teams tried to make context play a role in their models with certain success. For instance, team **newsSweeper** (TC:5) (Singh et al., 2020) used RoBERTa to obtain and to concatenate representations for both the propaganda snippet and its context (i.e., sentence). Team **SocCogCom** (TC:13) (Krishnamurthy et al., 2020) reduced the context to a window of three words before and after the propaganda snippet.

As in the SI subtask, a number of teams achieved sizable improvements when using various features. For instance, team **BPGC** (TC:18) (Patil et al., 2020) included TF.IDF vectors of words and character n-grams, topic modeling, and sentence-level polarity, among others, to their ensemble model that used BERT and logistic regression. Team **SocCogCom** (TC:13) (Krishnamurthy et al., 2020) integrated semantic-level emotional salience features from CrystalFeel (Gupta and Yang, 2018) and word-level psycholinguistic features from LIWC (Pennebaker et al., 2015). Team **CyberWallE** (TC:8) (Blaschke et al., 2020) added named entities, rhetorical, and question features, while taking special care of repetitions as part of a complex ensemble architecture. According to team **UNTLing** (TC:27) (Petee and Palmer, 2020), considering NEs is particularly useful for propaganda techniques such as *Loaded Language* and *Flag Waving* (e.g., the latter usually includes references to idealized entities) and VAD features were useful for emotion-related propaganda techniques such as *Appeal to fear/prejudice* and *Doubt*. Team **DiSaster** (TC:11) (Kaas et al., 2020) combined BERT with features including frequency of the fragment in the article and in the sentence it appears in, and the inverse uniqueness of words in a span. The goal of these features was to compensate the inability of BERT to deal with distant context, specifically to target the technique *Repetition*. Team **Solomon** (TC:4) also targeted *Repetition* by using dynamic least common sub-sequence, which they used to score the similarity between the fragment and the context. Then, the fragment was considered to be a repetition if the score was greater than a threshold heuristically set with respect to the length of the fragment.

Some other teams decided to perform a normalization of the input texts, thus trying to reduce the representation diversity. This was the case of team **DUTH** (TC:10) (Bairaktaris et al., 2020), which mapped certain words into classes using named entity recognition with focus on person names and gazetteers containing names and variations of names of countries (255 entries), religions (35 entries), political ideologies (23 entries), and slogans (41 entries). The recognized categories were replaced by the category name in the input, before passing the input to BERT.

As the class distribution for subtask TC is heavily skewed, some teams tried balancing techniques. For example, team **Inno** (TC:7) (Grigorev and Ivanov, 2020) experimented with undersampling (i.e. removing some examples from the bigger classes), team **syrapropa** (TC:20) applied a cost adjustment to their BERT-based model, and team **UMSIForeseer** (TC:17) (Jiang et al., 2020) used a mix of oversampling and undersampling, which they combined using a bagging ensemble.

Finally, some teams used an overriding strategy on top of the output of their supervised models. For example, team **CyberWallE** (TC:8) (Blaschke et al., 2020) performed a rule-based label post-processing, and team **syrapropa** (TC:20) applied syntactic rules based on part of speech.

5 Results and Discussion

5.1 Results on the Span Identification Subtask

Table 5 shows the performance of the participating systems both on the testing and on the development partitions on the SI subtask. The baseline for subtask SI is a simple system that randomly generates spans, by first selecting the starting character of a span and then its length. As mentioned in Section 4, practically all approaches relied on Transformers to produce representations and then plugged their output into a sequential model, at the token level. It is worth observing that only three of the top-5 systems on the development set appear also among the top-5 systems on the test set. Indeed, teams **syrapropa** and **PALI** felt down from positions 1 and 2 on the development set to positions 25 and 18 on the test set, which suggests possible overfitting. The performance for the final top-3 systems on the test partition —**Hitachi**, **ApplicaAI**, and **aschern**— reflects robust systems that seem to generalize much better.

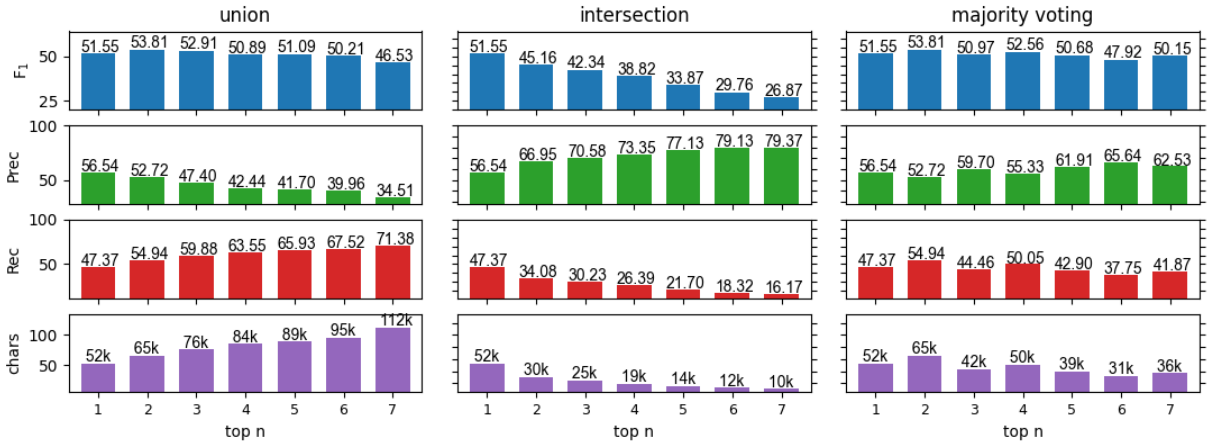


Figure 5: **SI Subtask**: performance (P, R, F_1) when combining the output of the top-7 systems using union, intersection, and majority voting. The bottom plots show the number of characters deemed propagandistic in each combination.

Figure 5 shows the performance evolution when combining the output of the top-performing systems on the test set. All operations are carried out at the character level, i.e., we label characters as propagandistic or not, and then we combine into spans the longest possible sequences of neighboring characters that we labeled as propagandistic. The union and the intersection use the corresponding set operations. In union, a character is considered propagandistic if at least one of the participating systems has recognized it as part of a propaganda span. In intersection, a character is considered propagandistic if all systems have included it as part of a propaganda span. For majority voting, we consider a character propagandistic if more than 50% of the participating systems had included it as part of a propaganda span. We can see in Figure 5 that the precision and the recall trends behave just as we expected: a lower precision (higher recall) is observed when more systems are combined with a union operation, and the opposite is true for the intersection. Despite the loss in terms of precision, computing the union of the top-$[2, 3]$ systems results in a better performance than what each of the top systems could achieve on its own. Such a combination gathers large ensembles of Transformer representations together with self-supervision to produce additional training data and boundary post-processing. If we are interested in a high-precision model, applying the intersection would make more sense, as it reaches a precision of 66.95 when combining the top-2 systems; however, this comes at the cost of a sizable lost of spans, which results in considerable drop in recall. The majority voting combination lies somewhere in between: keeping reasonable levels for both the precision and the recall.

5.2 Results on the Technique Classification Subtask

Table 6 shows the performance of the participating systems on the test set for the TC subtask, and Table 7 reports the results on the development set. The baseline system for subtask TC is a logistic regression classifier using one feature only: the length of the fragment. A similar pattern as for the SI subtask is observed: only two of the top-5 systems on the development set appear among the top-5 systems on the test set, which is a sign of possible overfitting for some of the systems. At the same time, systems that appeared to have a modest performance on the development set could eventually reach a higher position on test. For instance, team **Hitachi**, which was ranked 8th on the development set, ended up in the third position on the test set.

The tables further show the performance for each of the 14 propaganda techniques. In general, the systems show reasonably good performance when predicting *Loaded Language* and *Name Calling or Labeling*. These two classes are the most frequent ones, by a margin, and are also among the shortest ones on average (cf. Figure 3). On the other hand, techniques 13 (*Straw man, red herring*) and 14 (*Bandwagon, reduction ad hitlerum, whataboutism*) are among the hardest to identify. They are also among the least frequent ones.

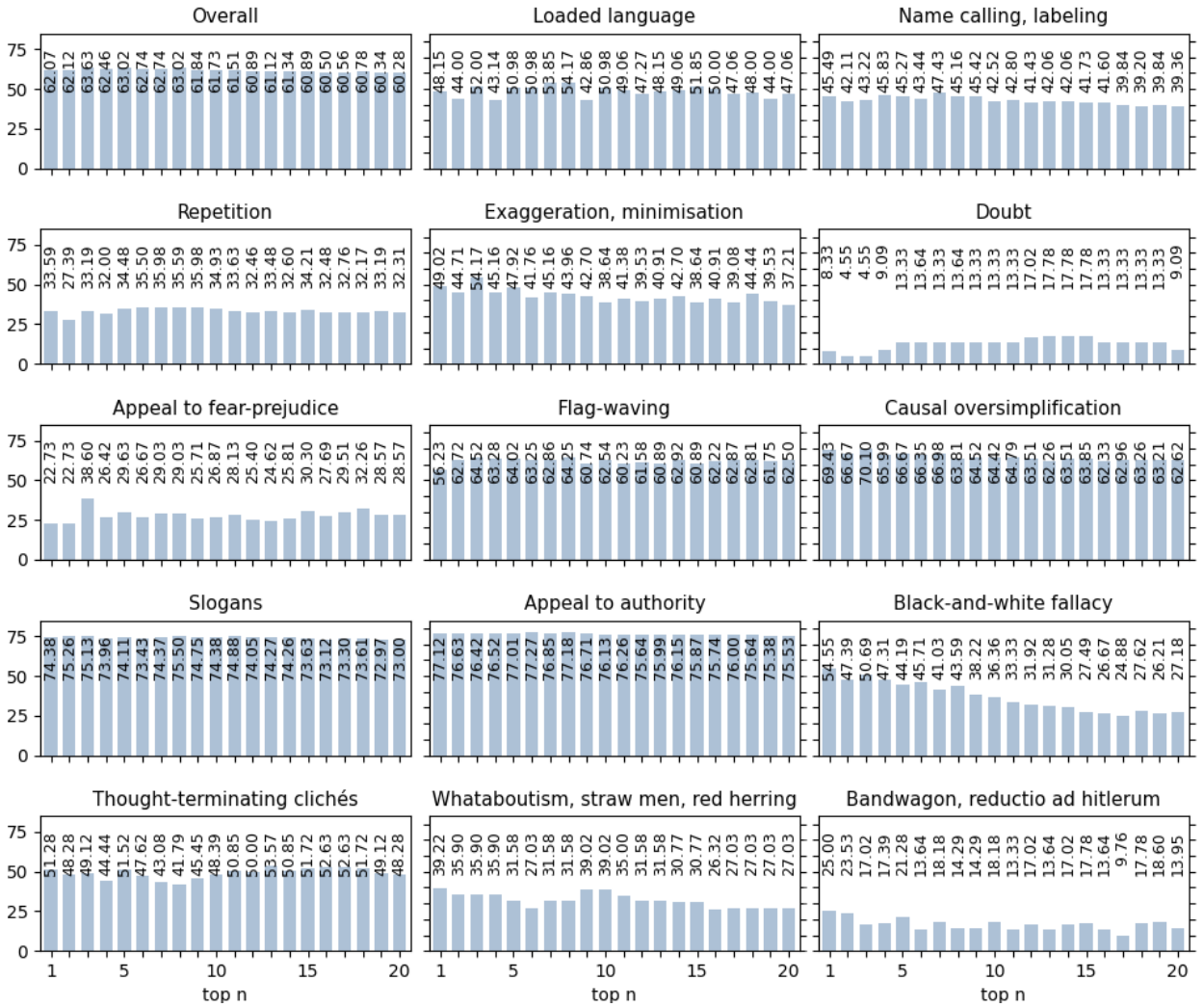


Figure 6: F_1 performance for the technique classification subtask when combining the top-20 systems with majority voting. The plots show the overall F_1 performance (top left) as well as the F_1 performance for each of the 14 propaganda techniques.

Once again, we studied the performance when combining more approaches. Figure 6 shows the performance evolution when combining different numbers of top-performing systems on the test set. As this is a multi-class problem, we combine the systems only on the basis of majority voting. In case of a tie, we prefer the more frequent propaganda technique on the training set. When looking at the overall picture, the performance evolution when adding more systems is fairly flat, reaching the top performance when combining the top-3 systems: an F_1 score of 63.63, which represents more than 1.5 points of absolute improvement over the top-1 system. When zooming into each of the fourteen propaganda techniques, we observe that in general the performance peak is indeed reached when considering three systems, e.g., for *Appeal to fear–prejudice*, *Exaggeration, minimisation*, or *Causal oversimplification*. Still for *Doubt*, which is the hardest class to recognize, as many as 13 systems are necessary in order to reach a (still discrete) peak performance of 17.78. Finally, note that there are other classes, such as *Black-and-white fallacy* or *Whatabaotism, straw men, red herring*, for which system combinations do not help.

	Test				Development			
Team	**Rnk**	**F_1**	**P**	**R**	**Rnk**	**F_1**	**P**	**R**
Hitachi	1	**51.55**	56.54	47.37	4	**50.12**	42.26	61.56
ApplicaAI	2	49.15	59.95	41.65	3	52.19	47.15	58.44
aschern	3	49.10	53.23	45.56	5	49.99	44.53	56.98
LTIatCMU	4	47.66	50.97	44.76	7	49.06	43.38	56.47
UPB	5	46.06	58.61	37.94	8	46.79	42.44	52.13
Fragarach	6	45.96	54.26	39.86	12	44.27	41.68	47.21
NoPropaganda	7	44.68	55.62	37.34	9	46.13	40.65	53.31
CyberWallE	8	43.86	42.16	45.70	17	42.39	33.45	57.86
Transformers	9	43.60	49.86	38.74	14	43.06	40.85	45.52
SWEAT	10	43.22	52.77	36.59	16	42.51	42.97	42.06
YNUtaoxin	11	43.21	55.62	35.33	11	44.35	40.74	48.67
DREAM	12	43.10	54.54	35.63	19	42.15	42.66	41.65
newsSweeper	13	42.21	46.52	38.63	10	44.45	38.76	52.10
PsuedoProp	14	41.20	41.54	40.87	22	39.32	34.27	46.11
Solomon	15	40.68	53.95	32.66	15	42.86	43.24	42.49
YNUHPCC	16	40.63	36.55	45.74	18	42.27	32.08	61.95
NLFIIT	17	40.58	50.91	33.73	21	39.67	35.04	45.72
PALI	18	40.57	53.20	32.79	2	52.35	49.64	55.37
UESTCICSA	19	39.85	56.09	30.90	13	44.17	43.21	45.18
TTUI	20	39.84	**66.88**	28.37	6	49.59	48.76	50.44
BPGC	21	38.74	49.39	31.88	25	36.79	34.72	39.12
DoNotDistribute	22	37.86	42.36	34.23	24	37.73	32.41	45.12
UTMNandOCAS	23	37.49	37.97	37.03	31	34.35	23.65	62.69
Entropy	24	37.23	41.68	33.63	32	32.89	30.82	35.25
syrapropa	25	36.20	49.53	28.52	1	53.40	39.88	80.80
SkoltechNLP	26	34.07	46.52	26.87	26	36.70	34.99	38.59
NTUAAILS	27	33.60	46.05	26.44	33	31.21	27.95	35.35
UAIC1860	28	33.21	24.49	51.57	34	30.27	20.69	56.37
CCNI	29	29.48	38.09	24.05	35	29.61	29.04	30.21
NCCU-SMRG	30	28.47	17.30	**80.37**	42	15.83	09.12	59.92
3218IR	31	23.47	22.63	24.38	40	20.03	15.10	29.76
WMD	32	20.09	47.11	12.77	27	36.34	33.26	40.05
LS	33	18.18	34.14	12.39	29	35.49	22.41	**85.33**
HunAlize	34	3.19	23.24	1.71	38	24.45	37.75	18.08
YOLO	35	0.72	17.20	0.37	46	0.64	9.36	0.33
Baseline	36	0.31	13.04	0.16	43	1.10	11.00	0.58
Murgila	–	–	–	–	20	41.38	32.96	55.56
TakeLab	–	–	–	–	23	39.06	38.85	39.27
atulcst	–	–	–	–	28	36.29	38.15	34.61
AAA	–	–	–	–	30	34.68	30.61	40.00
CUNLP	–	–	–	–	36	27.78	**58.23**	18.24
IIITD	–	–	–	–	37	25.82	18.81	41.15
UoB	–	–	–	–	39	24.02	22.30	26.04
UBirmingham	–	–	–	–	41	16.95	23.07	13.39
SocCogCom	–	–	–	–	44	0.79	9.97	0.41
Inno	–	–	–	–	45	0.64	9.36	0.33
Raghavan	–	–	–	–	47	0.40	0.20	33.45
California	–	–	–	–	48	0.39	5.92	0.20

Table 5: **Subtask 1: Span Identification (SI) performance on test and development.** The highest scores for each measure are highlighted. (*Note: We found a bug in the evaluation script after the end of the competition. The correct ranking, shown in Appendix B, does not differ substantially from above.*)

Rnk	Team	Overall	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	ApplicaAI	**62.07**	77.12	74.38	**54.55**	33.59	56.23	**45.49**	**69.43**	22.73	51.28	48.15	49.02	39.22	25.00	8.33
2	aschern	62.01	77.02	**75.65**	53.38	32.65	59.44	41.78	66.35	25.97	**54.24**	35.29	**53.57**	**42.55**	18.87	14.93
3	Hitachi	61.73	75.64	74.20	37.88	34.58	**63.43**	38.94	68.02	**36.62**	45.61	40.00	47.92	29.41	**26.92**	4.88
4	Solomon	58.94	74.66	70.75	42.53	28.44	61.82	39.39	61.84	19.61	50.75	26.67	42.00	38.10	0.00	4.88
5	newsSweeper	58.44	75.32	74.23	20.69	37.10	56.55	42.80	60.53	19.72	50.75	41.67	25.00	21.62	8.00	13.04
6	NoPropaganda	58.27	**77.17**	73.90	42.71	**37.99**	56.27	38.02	59.30	12.12	42.42	23.26	8.70	23.26	0.00	0.00
7	Inno	57.99	73.31	74.30	24.89	35.39	58.65	45.09	59.41	24.32	43.75	43.14	40.40	29.63	19.36	10.71
8	CyberWallE	57.37	74.68	70.92	47.68	28.34	58.65	39.84	54.38	15.39	39.39	14.63	23.68	23.81	0.00	12.25
9	PALI	57.32	74.29	69.09	24.56	28.57	58.97	36.59	61.62	30.59	39.22	27.59	39.62	40.82	20.90	**28.57**
10	DUTH	57.21	73.71	71.41	20.10	28.24	59.16	33.33	58.95	26.23	34.78	44.44	33.33	27.03	17.78	9.30
11	DiSaster	56.65	74.49	68.10	20.44	30.64	59.12	35.25	58.25	14.63	42.55	51.16	26.67	19.05	4.35	20.41
12	djichen	56.54	73.21	68.38	29.75	31.42	60.00	33.65	56.19	22.79	30.77	37.50	43.81	27.91	18.87	20.83
13	SocCogCom	55.81	72.18	67.34	18.88	34.86	60.40	31.62	54.26	6.35	40.91	28.57	26.51	23.53	10.00	9.76
14	TTUI	55.64	73.22	68.49	21.18	32.20	57.40	41.48	61.68	23.08	37.50	28.24	35.29	25.00	20.29	24.56
15	JUST	55.31	71.96	64.73	21.94	29.57	58.26	37.10	62.56	27.27	33.33	**48.89**	28.89	31.82	28.57	24.49
16	NLFIIT	55.25	72.55	69.30	21.55	30.30	55.66	24.89	63.32	0.00	41.67	29.63	32.10	13.64	0.00	9.30
17	UMSIForeseer	55.14	73.02	70.79	21.49	28.57	57.21	31.97	56.14	0.00	39.22	29.41	0.00	14.29	0.00	9.76
18	BPGC	54.81	71.58	67.51	23.74	33.47	53.78	33.65	58.93	24.18	40.00	30.77	40.00	20.69	20.90	12.50
19	UPB	54.30	70.09	68.86	20.00	30.62	52.55	30.00	55.87	16.95	34.62	20.00	19.72	22.86	4.88	0.00
20	syrapropa	54.25	71.47	68.44	30.77	28.10	56.14	29.77	57.02	21.51	29.03	31.58	30.61	28.57	9.09	19.61
21	WMD	52.01	69.33	64.67	13.89	25.46	53.94	29.20	52.08	5.71	6.90	7.14	0.00	7.41	0.00	5.00
22	YNUHPCC	50.50	68.08	62.33	17.72	21.54	51.04	26.40	55.56	3.45	27.59	29.79	38.38	17.78	15.00	13.79
23	UESTCICSA	49.94	68.23	66.88	27.96	25.44	44.99	22.75	53.14	3.74	41.38	12.77	11.27	28.57	3.70	0.00
24	DoNotDistribute	49.72	68.44	60.65	19.44	27.23	46.25	29.75	53.76	14.89	28.07	22.64	24.49	12.25	9.68	4.55
25	NTUAAILS	46.37	65.79	54.55	18.43	29.66	48.75	28.31	46.47	0.00	13.79	36.36	0.00	11.43	4.08	9.76
26	UAIC1860	41.17	62.33	42.97	11.16	21.01	36.41	22.12	38.78	7.60	11.43	17.39	2.90	5.56	4.26	9.76
27	UNTLing	39.11	62.57	36.74	7.78	11.82	32.65	5.29	40.48	2.86	17.65	4.35	0.00	0.00	0.00	0.00
28	HunAlize	37.10	58.59	15.82	2.09	23.81	31.76	11.83	29.95	7.84	4.55	6.45	8.00	0.00	0.00	0.00
29	Transformers	26.54	47.55	24.06	2.86	0.00	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	Baseline	25.20	46.48	0.00	19.26	14.42	29.14	3.68	6.20	11.56	0.00	0.00	0.00	0.00	0.00	0.00
31	Entropy	20.39	37.74	15.49	5.83	6.39	12.81	6.32	4.95	7.41	0.00	3.92	2.27	0.00	6.78	0.00
32	IJSE8	19.72	38.07	14.70	4.92	8.23	15.47	7.07	8.57	2.27	0.00	0.00	0.00	0.00	0.00	0.00

Table 6: **Technique classification F_1 performance on the test set**. The systems are ordered on the basis of the final ranking. Columns **1** to **14** show the performance for each of the propaganda techniques (cf. Section 2). The best score for each technique appears highlighted. (*Note: We found a bug in the evaluation script after the end of the competition. The correct ranking, shown in Appendix B, does not differ substantially from above.*)

Rnk	Team	Overall	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	ApplicaAI	**70.46**	80.42	74.11	70.67	**60.93**	63.57	**48.49**	**82.56**	47.06	62.65	**42.86**	20.00	43.48	**41.67**	80.00
2	aschern	68.11	80.00	73.85	67.39	59.36	**66.19**	45.24	74.25	30.44	63.89	28.57	32.43	33.33	38.46	40.00
8	Hitachi	65.19	77.99	71.16	46.98	56.58	62.96	36.96	81.40	50.00	67.61	12.50	22.22	19.05	40.91	75.00
16	Solomon	59.55	75.92	71.98	33.48	49.65	51.28	39.08	74.68	34.29	45.90	0.00	20.83	32.26	20.83	0.00
5	FakeSolomon	67.17	79.29	74.38	66.04	50.75	61.22	42.72	74.36	31.25	**70.00**	9.52	18.75	43.48	6.25	**88.89**
10	newsSweeper	62.75	76.23	71.35	45.14	55.32	56.94	44.90	74.29	47.83	60.53	16.67	6.90	16.67	6.45	57.14
11	NoPropaganda	60.68	77.06	72.49	47.69	38.71	50.36	33.33	75.74	48.00	53.17	0.00	0.00	8.33	0.00	0.00
13	Inno	60.11	76.57	70.62	34.67	43.75	52.86	40.82	77.97	34.15	59.70	11.77	35.71	18.18	15.39	61.54
7	CyberWallE	66.42	76.62	**81.00**	**73.29**	52.70	53.85	30.61	73.68	21.05	51.35	18.18	21.43	17.39	0.00	22.22
6	PALI	66.89	78.01	76.34	61.59	48.61	58.46	40.45	79.53	42.11	58.33	41.67	35.56	**45.71**	27.03	75.00
23	DUTH	57.86	76.12	70.32	23.08	46.15	48.72	34.41	64.43	42.86	55.39	9.52	11.11	15.39	11.11	50.00
9	DiSaster	62.84	76.73	77.33	47.11	48.28	56.21	35.29	75.15	23.08	35.09	30.00	0.00	9.09	5.56	28.57
26	djichen	57.57	76.04	70.53	24.89	44.60	48.28	36.78	76.36	31.82	51.43	13.33	20.00	10.81	4.55	0.00
28	SocCogCom	57.01	70.65	64.38	31.78	45.67	53.99	32.91	77.11	28.57	30.19	0.00	21.43	12.90	6.45	57.14
17	TTUI	58.98	75.08	71.00	32.07	41.96	56.92	34.69	77.53	33.33	50.00	16.22	19.05	20.69	29.09	66.67
24	JUST	57.67	74.97	72.36	23.64	49.64	48.75	31.07	73.75	23.81	29.03	0.00	0.00	0.00	5.88	0.00
20	NLFIIT	58.51	74.46	70.48	33.33	41.48	53.13	35.14	75.00	34.29	60.53	13.33	27.78	24.24	24.14	60.00
19	UMSIForeseer	58.70	74.44	70.95	31.76	51.80	47.95	31.58	75.86	**52.63**	43.33	10.00	0.00	0.00	10.00	0.00
18	BPGC	58.80	75.45	70.05	29.75	49.64	52.86	32.43	75.82	27.03	52.31	19.05	18.75	23.08	9.52	0.00
21	UPB	58.33	74.02	71.64	24.16	50.00	46.15	29.89	70.73	34.29	50.00	7.41	8.70	0.00	6.06	57.14
3	syrapropa	67.83	77.24	77.00	70.12	51.39	56.74	40.82	78.89	41.86	68.49	20.00	13.79	26.09	24.24	60.00
33	WMD	52.31	71.57	57.70	40.00	35.12	39.77	26.83	63.10	6.90	15.69	19.05	0.00	0.00	0.00	40.00
29	YNUHPCC	56.16	70.57	69.45	29.75	36.36	51.39	26.09	73.05	27.27	40.58	6.45	18.18	26.09	21.05	54.55
25	UESTCICSA	57.57	74.15	65.04	48.37	39.66	46.48	38.71	63.23	26.67	58.82	0.00	0.00	36.36	9.52	0.00
30	DoNotDistribute	54.00	71.20	67.33	29.38	31.93	45.95	25.58	72.61	25.00	28.99	0.00	0.00	22.86	9.76	0.00
32	NTUAAILS	53.25	69.76	59.90	27.19	43.64	51.61	21.33	73.03	17.14	27.12	6.67	7.14	0.00	8.00	44.44
34	UAIC1860	43.84	57.88	40.37	14.12	23.66	42.03	7.02	60.00	0.00	4.55	10.53	0.00	0.00	0.00	33.33
37	UNTLing	40.92	59.45	33.33	11.83	12.28	35.42	11.94	51.70	26.67	25.00	8.33	0.00	0.00	5.88	0.00
36	HunAlize	41.02	56.13	40.75	5.32	22.22	41.18	9.23	53.50	33.33	0.00	0.00	0.00	0.00	0.00	0.00
41	Transformers	30.10	48.50	28.62	12.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.88	0.00
43	Baseline	26.53	40.58	0.00	38.50	11.68	19.20	9.38	8.28	7.23	0.00	0.00	0.00	0.00	0.00	0.00
27	Entropy	57.10	75.22	66.50	28.19	45.16	53.60	31.11	74.85	29.27	46.88	13.33	18.18	20.00	10.00	0.00
4	FLZ	67.17	**81.09**	71.12	70.78	23.38	65.46	20.00	64.08	46.15	64.00	44.44	**37.04**	28.57	6.67	75.00
12	Fragarach	60.49	77.12	70.50	38.21	50.69	51.03	40.91	74.12	34.04	45.71	14.29	12.90	16.67	6.06	57.14
14	NerdyBirdies	59.83	76.23	73.55	33.05	48.65	52.06	36.04	70.30	47.62	48.57	15.39	8.70	20.00	13.33	0.00
15	CUNLP	59.55	74.38	73.30	47.62	43.66	50.69	34.29	71.90	35.90	55.07	16.22	15.79	19.51	8.89	25.00
22	Tianyi	58.23	75.39	71.22	26.43	44.72	50.36	34.57	75.61	28.57	49.32	13.33	11.77	0.00	0.00	57.14
31	Murgila	53.53	71.21	66.14	24.55	42.55	37.29	22.47	69.51	30.00	40.58	17.65	13.33	31.58	22.64	44.44
35	hseteam	41.40	63.65	42.82	22.40	32.26	26.29	11.11	57.52	7.69	26.67	0.00	7.27	0.00	4.76	42.86
38	HenryAtDuderstadt	34.15	52.97	4.10	0.00	0.00	34.29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
39	XJPaccelerationMASTER	32.46	48.76	0.00	0.00	0.00	33.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	SWEAT	30.57	46.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
42	NCCU-SMRG	29.26	37.92	34.46	25.53	14.69	36.92	15.09	45.07	9.76	13.33	12.50	16.67	12.50	0.00	23.08
44	LS	26.23	39.20	0.00	39.31	12.95	19.67	9.09	8.28	8.79	0.00	0.00	0.00	6.25	0.00	0.00
45	SkoltechNLP	26.23	39.20	0.00	39.31	12.95	19.67	9.09	8.28	8.79	0.00	0.00	0.00	6.25	0.00	0.00
46	TakeLab	25.31	42.92	24.70	1.37	0.00	6.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
47	UTMNandOCAS	23.42	40.53	19.74	18.67	4.44	0.00	10.53	0.00	0.00	0.00	0.00	23.08	8.70	0.00	0.00

Table 7: **Technique classification F_1 performance on the development set**. The systems are ordered based on the final ranking on the test set (cf. Table 6), whereas the ranking is the one on the development set. Columns **1** to **14** show the performance on each class (cf. Section 2). The best score for each class is bold.

6 Related Work

Propaganda is particularly visible in the context of "fake news" on social media, which have attracted a lot of research recently (Shu et al., 2017). Thorne and Vlachos (2018) surveyed fact-checking approaches to fake news and related problems, and Li et al. (2016) focused on truth discovery in general. Two recent articles in *Science* offered a general discussion on the science of "fake news" (Lazer et al., 2018) and the process of proliferation of true and false news online (Vosoughi et al., 2018).

We are particularly interested here in how different forms of propaganda are manifested in text. So far, the computational identification of propaganda has been tackled mostly at the article level. Rashkin et al. (2017) created a corpus, where news articles are labeled as belonging to one of the following four categories: *propaganda*, *trusted*, *hoax*, or *satire*. The articles came from eight sources, two of which were propagandistic. The labels were obtained using distant supervision, assuming that all articles from a given news source share the label of that source, which introduces noise (Horne et al., 2018). Barrón-Cedeño et al. (2019b) experimented with a binary version of the problem: *propaganda* vs. *no propaganda*. See (Da San Martino et al., 2020a) for a recent survey on computational propaganda detection.

In general, propaganda techniques serve as a means to persuade people, often in argumentative settings. While they may increase the rhetorical effectiveness of arguments, they naturally harm other aspects of argumentation quality (Wachsmuth et al., 2017). In particular, many of the span propaganda techniques considered in this shared task relate to the notion of *fallacies*, i.e. arguments whose reasoning is flawed in some way, often hidden and often on purpose (Tindale, 2007). Some recent work in computational argumentation has dealt with such fallacies. Among these, Habernal et al. (2018) presented and analyzed a corpus of web forum discussions with *Ad hominem* fallacies, and Habernal et al. (2017) introduced *Argotario*, a game that educates people to recognize fallacies. Argotario also had a corpus as a by-product, with 1.3k arguments annotated for five fallacies, including *Ad hominem*, *Red herring*, and *Irrelevant authority*, which are related to some of our propaganda techniques (cf. Section 2). Unlike these corpora, the news articles in our corpus are annotated with fourteen propaganda techniques. Moreover, instead of labeling entire arguments, our annotation aims at identifying the minimal text spans related to a technique.

In the present SemEval task, we departed from the eighteen propaganda techniques and the corpus described in (Da San Martino et al., 2019b; Yu et al., 2019).[8] We used the news articles included in that corpus in a pilot task that ran in January 2019, the *Hack the News Datathon*,[9] as well as in a previous shared task, held as part of the *2019 Workshop on NLP4IF: Censorship, Disinformation, and Propaganda*.[10] Both the datathon and the shared task tackled the identification of propaganda techniques as one overall task (along with a binary sentence-level propaganda classification task), i.e. without splitting it into subtasks as we did here. As detailed in the overview paper of Da San Martino et al. (2019a), the best-performing systems in the shared task used BERT-based contextual representations. Other systems used contextual representations based on RoBERTa, Grover, and ELMo, or context-independent representations based on lexical, sentiment, readability, and TF-IDF features. As in the task at hand, ensembles were also popular. Still, the most successful submissions achieved an F_1-score of 24.88 only (and only 10.43 in the datathon). This is why, here we decided to split the task into subtasks in order to allow researchers to focus on one subtask at a time. Moreover, we merged some of the original 18 propaganda techniques to reduce data sparseness issues.

Other related shared tasks include the FEVER 2018 and 2019 tasks on *Fact Extraction and VERification* (Thorne et al., 2018), the SemEval 2017 and 2019 tasks on determining the veracity of rumors (Derczynski et al., 2017; Gorrell et al., 2019) and the SemEval 2019 task on *Fact-Checking in Community Question Answering Forums* (Mihaylova et al., 2019). Also, the CLEF 2018–2020 *CheckThat!* labs' shared tasks (Nakov et al., 2018; Elsayed et al., 2019a; Elsayed et al., 2019b; Barrón-Cedeño et al., 2020a; Barrón-Cedeño et al., 2020b), which featured tasks on automatic identification (Atanasova et al., 2018; Atanasova et al., 2019) and verification (Barrón-Cedeño et al., 2018; Hasanain et al., 2019; Hasanain et al., 2020; Shaar et al., 2020) of claims in political debates and in social media.

[8] You can also try the `Prta` system (Da San Martino et al., 2020b) online at: `http://www.tanbih.org/prta`

[9] `http://www.datasciencesociety.net/hack-news-datathon/`

[10] `http://www.netcopia.net/nlp4if/2019/`

7 Conclusion and Future Work

We have described SemEval-2020 Task 11 on Detection of Propaganda Techniques in News Articles. The task attracted the interest of a number of researchers: 250 teams signed up to participate, and 44 made submissions on the test dataset. We received 35 and 31 submissions for subtask SI and subtask TC, respectively. Overall, subtask SI (segment identification) was easier and all systems managed to improve over the baseline. However, subtask TC (technique classification) proved to be much more challenging, and some teams could not improve over our baseline.

In future work, we plan to extend the dataset to cover more examples as well as more propaganda techniques. We further plan to develop similar datasets for other languages.

Acknowledgments

We thank the anonymous reviewers for their constructive comments and suggestions, which have helped us improve the final version of this paper. We further thank Anton Chernyavskiy for pointing us to the bug in the evaluation script.

The task is organized within the Propaganda Analysis Project,[11] part of the Tanbih project.[12] Tanbih aims to limit the effect of "fake news", propaganda, and media bias by making users aware of what they are reading, thus promoting media literacy and critical thinking.

References

Ola Altiti, Malak Abdullah, and Rasha Obiedat. 2020. JUST at SemEval-2020 Task 11: Detecting propaganda techniques using BERT pretrained model. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Anastasios Arsenos and Georgios Siolas. 2020. NTUAAILS at SemEval-2020 Task 11: Propaganda detection and classification with biLSTMs and ELMo. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Pepa Atanasova, Lluís Màrquez, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Wajdi Zaghouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the CLEF-2018 CheckThat! lab on automatic identification and verification of political claims, task 1: Check-worthiness. In *CLEF 2018 Working Notes*, Avignon, France.

Pepa Atanasova, Preslav Nakov, Georgi Karadzhov, Mitra Mohtarami, and Giovanni Da San Martino. 2019. Overview of the CLEF-2019 CheckThat! Lab on automatic identification and verification of claims. Task 1: Check-worthiness. In *CLEF 2019 Working Notes*, Lugano, Switzerland.

Anastasios Bairaktaris, Symeon Symeonidis, and Avi Arampatzis. 2020. DUTH at SemEval-2020 Task 11: BERT with entity mapping for propaganda classification. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Pepa Atanasova, Wajdi Zaghouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the CLEF-2018 CheckThat! lab on automatic identification and verification of political claims, task 2: Factuality. In *CLEF 2018 Working Notes*, Avignon, France.

Alberto Barrón-Cedeño and Paolo Rosso. 2009. On automatic plagiarism detection based on n-grams comparison. In *Proceedings of the 31th European Conference on IR Research*, ECIR '09, pages 696–700, Toulouse, France.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019a. Proppy: A system to unmask propaganda in online news. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, AAAI'19, pages 9847–9848, Honolulu, HI, USA.

Alberto Barrón-Cedeño, Israa Jaradat, Giovanni Da San Martino, and Preslav Nakov. 2019b. Proppy: Organizing the news based on their propagandistic content. *Information Processing and Management*, 56(5):1849–1864, September.

[11] `http://propaganda.qcri.org`
[12] `http://tanbih.qcri.org`

Alberto Barrón-Cedeño, Tamer Elsayed, Preslav Nakov, Giovanni Da San Martino, Maram Hasanain, Reem Suwaileh, Fatima Haouari, Nikolay Babulkov, Bayan Hamdan, Alex Nikolov, Shaden Shaar, and Zien Sheikh Ali. 2020a. Overview of CheckThat! 2020 — automatic identification and verification of claims in social media. In *Proceedings of the 11th International Conference of the CLEF Association: Experimental IR Meets Multilinguality, Multimodality, and Interaction*, CLEF '2020, Thessaloniki, Greece.

Alberto Barrón-Cedeño, Tamer Elsayed, Preslav Nakov, Giovanni Da San Martino, Maram Hasanain, Reem Suwaileh, and Fatima Haouari. 2020b. CheckThat! at CLEF 2020: Enabling the automatic identification and verification of claims in social media. In *Proceedings of the 42nd European Conference on Information Retrieval*, ECIR '20, pages 499–507, Lisbon, Portugal.

Verena Blaschke, Maxim Korniyenko, and Sam Tureski. 2020. CyberWallE at SemEval-2020 Task 11: An analysis of feature engineering for ensemble models for propaganda detection. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Bolsover and Howard. 2017. Computational propaganda and political big data: Toward a more critical research agenda. *Big Data*, 5(4):273–276.

Akemi Takeoka Chatfield, Christopher G. Reddick, and Uuf Brajawidagda. 2015. Tweeting propaganda, radicalization and recruitment: Islamic state supporters multi-sided Twitter networks. In *Proceedings of the 16th Annual International Conference on Digital Government Research*, pages 239–249, Phoenix, AZ, USA.

Aniruddha Chauhan and Harshita Diddee. 2020. PsuedoProp at SemEval-2020 Task 11: Propaganda span detection using BERT-CRF and ensemble sentence level classifier. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Wei-Te Chen and Will Styler. 2013. Anafora: A web-based general purpose annotation tool. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Demonstration Session)*, NAACL-HLT '13, pages 14–19, Atlanta, GA, USA.

Anton Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. 2020. aschern at SemEval-2020 Task 11: It takes three to tango: RoBERTa, CRF, and transfer learning. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Giovanni Da San Martino, Alberto Barrón-Cedeño, and Preslav Nakov. 2019a. Findings of the NLP4IF-2019 shared task on fine-grained propaganda detection. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, NLP4IF '19, pages 162–170, Hong Kong, China.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019b. Fine-grained analysis of propaganda in news articles. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, EMNLP-IJCNLP '19, pages 5636–5646, Hong Kong, China.

Giovanni Da San Martino, Stefano Cresci, Alberto Barrón-Cedeño, Seunghak Yu, Roberto Di Pietro, and Preslav Nakov. 2020a. A survey on computational propaganda detection. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence*, IJCAI-PRICAI '20, pages 4826–4832, Yokohama, Japan.

Giovanni Da San Martino, Shaden Shaar, Yifan Zhang, Seunghak Yu, Alberto Barrón-Cedeño, and Preslav Nakov. 2020b. Prta: A system to support the analysis of propaganda techniques in the news. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, ACL '20, pages 287–293.

Lavinia Dan. 2015. Techniques for the translation of advertising slogans. In *Proceedings of the International Conference Literature, Discourse and Multicultural Dialogue*, pages 13–23, Tirgu-Mures, Mures.

Jiaxu Dao, Jin Wang, and Xuejie Zhang. 2020. YNU-HPCC at SemEval-2020 Task 11: LSTM network for detection of propaganda techniques in news articles. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Guillaume Daval-Frerot and Weis Yannick. 2020. WMD at SemEval-2020 Tasks 7 and 11: Assessing humor and propaganda using unsupervised data augmentation. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Daryna Dementieva, Igor Markov, and Alexander Panchenko. 2020. SkoltechNLP at SemEval-2020 Task 11: Exploring unsupervised text augmentation for propaganda detection. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, pages 60–67, Vancouver, Canada.

Dimas Sony Dewantara, Indra Budi, and Muhammad Okky Ibrohim. 2020. 3218IR at SemEval-2020 Task 11: Conv1D and word embedding in propaganda span identification at news articles. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Ilya Dimov, Vladislav Korzun, and Ivan Smurov. 2020. NoPropaganda at SemEval-2020 Task 11: A borrowed approach to sequence tagging and text classification. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Tamer Elsayed, Preslav Nakov, Alberto Barrón-Cedeño, Maram Hasanain, Reem Suwaileh, Giovanni Da San Martino, and Pepa Atanasova. 2019a. CheckThat! at CLEF 2019: Automatic identification and verification of claims. In *Proceedings of the 41st European Conference on Information Retrieval*, ECIR '19, pages 309–315, Cologne, Germany.

Tamer Elsayed, Preslav Nakov, Alberto Barrón-Cedeño, Maram Hasanain, Reem Suwaileh, Giovanni Da San Martino, and Pepa Atanasova. 2019b. Overview of the CLEF-2019 CheckThat!: Automatic identification and verification of claims. In *Proceedings of the 10th International Conference of the CLEF Association*, CLEF '19, pages 301–321, Lugano, Switzerland.

Vlad Ermurachi and Daniela Gifu. 2020. UAIC1860 at SemEval-2020 Task 11: Detection of propaganda techniques in news articles. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Lei Gao, Alexis Kuppersmith, and Ruihong Huang. 2017. Recognizing explicit and implicit hate speech using a weakly supervised two-path bootstrapping approach. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, IJCNLP '17, pages 774–782, Taipei, Taiwan.

Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. 2015. Quantifying controversy in social media. *ACM Transactions on Social Computing*, 1(1):1–27, jan.

Monika Glowacki, Vidya Narayanan, Sam Maynard, Gustavo Hirsch, Bence Kollanyi, Lisa-Maria Neudert, Phil Howard, Thomas Lederer, and Vlad Barash. 2018. News and political information consumption in Mexico: Mapping the 2018 Mexican presidential election on Twitter and Facebook. Technical Report COMPROP DATA MEMO 2018.2, Oxford University, Oxford, UK, June.

Jean Goodwin. 2011. Accounting for the force of the appeal to authority. In *Proceedings of the 9th International Conference of the Ontario Society for the Study of Argumentation*, pages 1–9, Windsor, Canada.

Genevieve Gorrell, Ahmet Aker, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2019. SemEval-2019 task 7: RumourEval, determining rumour veracity and support for rumours. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, SemEval '19, pages 845–854, Minneapolis, MN, USA.

Dmitry Grigorev and Vladimir Ivanov. 2020. Inno at SemEval-2020 Task 11: Leveraging pure transfomer for multi-class propaganda detection. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Raj Kumar Gupta and Yinping Yang. 2018. CrystalFeel at SemEval-2018 task 1: Understanding and detecting emotion intensity using affective lexicons. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, SemEval '18, pages 256–263, New Orleans, LA, USA.

Ivan Habernal, Raffael Hannemann, Christian Pollak, Christopher Klamm, Patrick Pauli, and Iryna Gurevych. 2017. Argotario: Computational argumentation meets serious games. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, EMNLP '17, pages 7–12, Copenhagen, Denmark.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018. Before name-calling: Dynamics and triggers of ad hominem fallacies in web argumentation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 386–396, New Orleans, LA, USA.

Maram Hasanain, Reem Suwaileh, Tamer Elsayed, Alberto Barrón-Cedeño, and Preslav Nakov. 2019. Overview of the CLEF-2019 CheckThat! Lab on automatic identification and verification of claims. Task 2: Evidence and factuality. In *CLEF 2019 Working Notes*, Lugano, Switzerland.

Maram Hasanain, Fatima Haouari, Reem Suwaileh, Zien Sheikh Ali, Bayan Hamdan, Tamer Elsayed, Alberto Barrón-Cedeño, Giovanni Da San Martino, and Preslav Nakov. 2020. Overview of CheckThat! 2020 Arabic: Automatic identification and verification of claims in social media. In *Working Notes of CLEF 2020*, CLEF '2020, Thessaloniki, Greece.

Renee Hobbs and Sandra Mcgee. 2008. Teaching about Propaganda: An Examination of the Historical Roots of Media Literacy. *Journal of Media Literacy Education*, 6(62):56–67.

Benjamin D. Horne, Sara Khedr, and Sibel Adali. 2018. Sampling the news producers: A large news and feature data set for the study of the complex media landscape. In *Proceedings of the Twelfth International Conference on Web and Social Media*, ICWSM '18, pages 518–527, Stanford, CA, USA.

John Hunter. 2015. Brainwashing in a large group awareness training? The classical conditioning hypothesis of brainwashing. Master's thesis, University of Kwazulu-Natal, South Africa.

Ins. 1938. How to detect propaganda. In *Propaganda Analysis. Volume I*, chapter 2, pages 210–218. Publications of the Institute for Propaganda Analysis, New York, NY, USA.

Yunzhe Jiang, Cristina Gârbacea, and Qiaozhu Mei. 2020. UMSIForeseer at SemEval-2020 Task 11: Propaganda detection by fine-tuning BERT with resampling and ensemble learning. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Garth S. Jowett and Victoria O'Donnell. 2012a. *Propaganda and Persuasion*. SAGE, Los Angeles, CA, USA, 5th edition.

Garth S. Jowett and Victoria O'Donnell. 2012b. What is propaganda, and how does it differ from persuasion? In *Propaganda and Persuasion*, chapter 1, pages 1–48. Sage Publishing.

Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. 2020. ApplicaAI at SemEval-2020 Task 11: On RoBERTa-CRF, Span CLS and whether self-training helps them. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Anders Friis Kaas, Viktor Torp Thomsen, and Barbara Plank. 2020. Team DiSaster at SemEval-2020 Task 11: Combining BERT and hand-crafted features for identifying propaganda techniques in news. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Sopan Khosla, Rishabh Joshi, Ritam Dutt, Alan W. Black, and Yulia Tsvetkov. 2020. LTIatCMU at SemEval-2020 Task 11: Incorporating multi-level features for multi-granular propaganda span identification. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Moonsung Kim and Steven Bethard. 2020. TTUI at SemEval-2020 Task 11: Propaganda detection with transfer learning and ensembles. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Michael Kranzlein, Shabnam Behzad, and Nazli Goharian. 2020. Team DoNotDistribute at SemEval-2020 Task 11: Features, finetuning, and data augmentation in neural models for propaganda detection in news articles. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Gangeshwar Krishnamurthy, Raj Kumar Gupta, and Yinping Yangi. 2020. SocCogCom at SemEval-2020 Task 11: Detecting propaganda techniques in news articles using semantic-level emotional salience features. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

David M.J. Lazer, Matthew A. Baum, Yochai Benkler, Adam J. Berinsky, Kelly M. Greenhill, Filippo Menczer, Miriam J. Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven A. Sloman, Cass R. Sunstein, Emily A. Thorson, Duncan J. Watts, and Jonathan L. Zittrain. 2018. The science of fake news. *Science*, 359(6380):1094–1096.

Jinfen Li and Lu Xiao. 2020. HybridPro at SemEval-2020 Task 11: A hybrid model for propagandistic technique classification in news articles. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. A survey on truth discovery. *SIGKDD Explor. Newsl.*, 17(2):1–16, February.

Matej Martinkovic, Samuel Pecar, and Marian Simko. 2020. NLFIIT at SemEval-2020 Task 11: Neural network architectures for detection of propaganda techniques in news articles. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Yann Mathet, Antoine Widlöcher, and Jean-Philippe Métivier. 2015. The unified and holistic method gamma (γ) for inter-annotator agreement measure and alignment. *Computational Linguistics*, 41(3):437–479, sep.

Tsvetomila Mihaylova, Georgi Karadzhov, Pepa Atanasova, Ramy Baly, Mitra Mohtarami, and Preslav Nakov. 2019. SemEval-2019 task 8: Fact checking in community question answering forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, SemEval '19, pages 860–869, Minneapolis, MN, USA.

Elena Mikhalkova, Nadezhda Ganzherli, Anna Glazkova, and Yulia Bidulia. 2020. UTMN at SemEval-2020 task 11: A kitchen solution to automatic propaganda detection. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Clyde R. Miller. 1939. The Techniques of Propaganda. From "How to Detect and Analyze Propaganda," an address given at Town Hall. The Center for learning.

Gaku Morio, Terufumi Morishita, Hiroaki Ozaki, and Toshinori Miyoshi. 2020. Hitachi at SemEval-2020 Task 11: An empirical study of pre-trained transformer family for propaganda detection. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Robert Muller. 2018. Indictment of Internet Research Agency. pages 1–37.

Preslav Nakov, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Wajdi Zaghouani, Pepa Atanasova, Spas Kyuchukov, and Giovanni Da San Martino. 2018. Overview of the CLEF-2018 CheckThat! lab on automatic identification and verification of political claims. In *Proceedings of the Ninth International Conference of the CLEF Association: Experimental IR Meets Multilinguality, Multimodality, and Interaction*, CLEF '18, pages 372–387, Avignon, France.

Andrei Paraschiv and Dumitru-Clementin Cercel. 2020. UPB at SemEval-2020 Task 11: Propaganda detection with domain-specific trained BERT. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Rajaswa Patil, Somesh Singh, and Swati Agarwal. 2020. BPGC at SemEval-2020 Task 11: Propaganda detection in news articles with multi-granularity knowledge sharing and linguistic features based ensemble learning. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

James W. Pennebaker, Ryan L. Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of LIWC2015.

Maias Petee and Alexis Palmer. 2020. UNTLing at SemEval-2020 Task 11: Detection of propaganda techniques in news articles. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Martin Potthast, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. 2010. An evaluation framework for plagiarism detection. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 997–1005, Beijing, China.

Mayank Raj, Ajay Jaiswal, Rohit R.R, Ankita Gupta, Sudeep Sahoo, Vertika Srivastava, and Kim Yeon Hyang. 2020. Solomon at SemEval-2020 Task 11: Novel ensemble architechture for fine-tuned propaganda detection in news articles. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP 17, pages 2931–2937, Copenhagen, Denmark.

Monika L. Richter. 2017. The Kremlin's platform for 'useful idiots' in the West: An overview of RT's editorial strategy and evidence of impact. *Kremlin Watch*, page 53.

Shaden Shaar, Alex Nikolov, Nikolay Babulkov, Firoj Alam, Alberto Barrón-Cedeño, Tamer Elsayed, Maram Hasanain, Reem Suwaileh, Fatima Haouari, Giovanni Da San Martino, and Preslav Nakov. 2020. Overview of CheckThat! 2020 English: Automatic identification and verification of claims in social media. In *Working Notes of CLEF 2020*, CLEF '2020, Thessaloniki, Greece.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36, September.

Paramansh Singh, Siraj Sandhu, Subham Kumar, and Ashutosh Modi. 2020. newsSweeper at SemEval-2020 Task 11: Context-aware rich feature representations for propaganda classification. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Xin Tao and Xiaobing Zhou. 2020. YNUtaoxin at SemEval-2020 Task 11: Identification fragments of propaganda technique by neural sequence labeling models with different tagging schemes and pre-trained language model. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Cristina Tardáguila, Fabrício Benevenuto, and Pablo Ortellado. 2018. Fake news is poisoning brazilian politics. WhatsApp can stop it. `https://www.nytimes.com/2018/10/17/opinion/brazil-election-fake-news-whatsapp.html`.

Gabriel H Teninbaum. 2009. Reductio ad hitlerum: Trumping the judicial Nazi card. *Michigan State Law Review*, 2009.

James Thorne and Andreas Vlachos. 2018. Automated fact checking: Task formulations, methods and future directions. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 3346–3359, Santa Fe, NM, USA.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 809–819, New Orleans, LA, USA.

Christopher W. Tindale. 2007. *Fallacies and Argument Appraisal. Critical Reasoning and Argumentation*. Cambridge University Press.

Robyn Torok. 2015. Symbiotic radicalisation strategies: Propaganda tools and neuro linguistic programming. In *Proceedings of the Australian Security and Intelligence Conference*, pages 58–65, Perth, Australia.

Ekansh Verma, Vinodh Motupalli, and Souradip Chakraborty. 2020. Transformers at SemEval-2020 Task 11: Propaganda fragment detection using diversified bert architectures based ensemble learning. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval '20, Barcelona, Spain.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst, and Benno Stein. 2017. Computational argumentation quality assessment in natural language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '17, pages 176–187, Valencia, Spain.

Douglas Walton. 2013. The straw man fallacy. In *Methods of Argumentation*, pages 249–286. Cambridge University Press.

Anthony Weston. 2000. *A Rulebook for Arguments*. Hackett Student Handbooks.

Seunghak Yu, Giovanni Da San Martino, and Preslav Nakov. 2019. Experiments in detecting persuasion techniques in the news. In *Proceedings of the NeurIPS 2019 Joint Workshop on AI for Social Good*, NeurIPS '19, Vancouver, Canada, December.

A Summary of all Submitted Systems

This appendix includes a brief summary of all systems for both subtasks. We present the teams in alphabetical order. The subindex on the right of each team represents its official test rank in the subtasks. The teams appearing in Tables 5, 6, or 7 but not here did not submit a paper describing their approach.

Team 3218IR (Dewantara et al., 2020) (SI:31) used a one-dimensional convolutional neural network (CNN) with word embeddings, whose number of layers and filters as well as kernel and pooling sizes were all tuned empirically.

Team ApplicaAI (Jurkiewicz et al., 2020) (SI: 2, TC: 1) applied self-supervision using the RoBERTa model. For the SI subtask, they used a RoBERTa-CRF architecture. The model trained using this architecture was then iteratively used to produce silver data by predicting on 500k sentences and retraining the model with both gold and silver data. As for subtask TC, ApplicaAI opted for feeding their models with propagandas snippets in context. Full sentences are shaped as the input with the specific propaganda in them. Once again, silver data was used, taking advantage of the spans detected by their SI model and labeling with their preliminary TC model. The final classifier was an ensemble of models trained on the original corpus, re-weighting, and a model trained also on silver data.

Team aschern (Chernyavskiy et al., 2020) (SI: 3, TC: 2) tackled both subtasks. For SI, they fine-tuned an ensemble of two differently intialized RoBERTa models, each with an attached CRF for sequence labeling and simple span character boundary post-processing. A RoBERTa ensemble was also used for TC, treating the task as sequence classification but using an average embedding of the surrounding tokens and the length of a span as contextual features. They further used transfer learning, to pass knowledge about the SI subtask to help the TC subtask. Finally, specific postprocessing was done to increase the consistency of the repetition technique spans and to avoid insertions of techniques in other techniques.

Team BPGC (Patil et al., 2020) (SI: 21, TC: 18) used a multi-granularity approach to address subtask SI. Information about the article and the sentence was considered when classifying each word as propaganda or not, by means of computing and concatenating vectorial representations for the three inputs. For subtask TC, they used an ensemble of BERT and logistic regression, complemented with engineered features which, as stated by the authors, were particularly useful for the smaller classes. Such features include TF.IDF vectors of words and character n-grams, topic modeling, and sentence-level polarity, among others. Different learning models were explored for both tasks, including LSTM and CNN, together with diverse Transformers to build ensembles of classifiers.

Team CyberWallE (Blaschke et al., 2020) (SI: 8, TC: 8) used BERT embeddings for subtask SI, as well as manual features modeling sentiment, rhetorical structure, and POS tags, which were eventually fed into a bi-LSTM to produce IO labels, followed by some post-processing to merge neighboring spans. For subtask TC, they extracted the pre-softmax layer of BERT and further added extra features (rhetorical, named entities, question), while taking special care of repetitions as part of a complex ensemble architecture, followed by label post-processing.

Team DiSaster (Kaas et al., 2020) (TC:11) used a combination of BERT and hand-crafted features, including frequency of the fragment in the article and in the sentence it appears in and the inverse uniqueness of words in a span. The goal of the features is to compensate the inability of BERT to deal with distant context, specifically to target the technique *Repetition*.

Team DoNotDistribute (Kranzlein et al., 2020) (SI: 22, TC: 24) opted for a combination of BERT-based models and engineered features (e.g., PoS, NEs, frequency within propaganda snippets in the training set). A reported performance increase of 5% was obtained by producing 3k new silver training instances. A library was used to create near-paraphrases of the propaganda snippets by randomly substituting certain PoS words.

Team DUTH (Bairaktaris et al., 2020) (TC:10) pre-processed the input including URL normalization, number and punctuation removal, as well as lowercasing. They further mapped certain words into classes using named entity recognition with focus on person names and gazetteers containing names and variations of names of countries (255 entries), religions (35 entries), political ideologies (23 entries), and slogans (41 entries). The recognized categories were replaced by the category name in the input, before passing the input to BERT.

Team Hitachi (Morio et al., 2020) (SI: 1, TC: 3) used BIO encoding for subtask SI, which is typical for related segmentation and labeling tasks such as named entity recognition. They used a complex heterogeneous multi-layer neural network, trained end-to-end. The network used a pre-trained language model, which generates a representation for each input token. To this were added part-of-speech (PoS) and named entity (NE) embeddings. As a result, there were three representations for each token, which were concatenated and used as an input to bi-LSTMs. At this moment, the network branches as it is trained with three objectives: (*i*) the main BIO tag prediction objective, and two auxiliary objectives, namely (*ii*) token-level technique classification, and (*iii*) sentence-level classification. There is one Bi-LSTM for objectives (*i*) and (*ii*), and there is another Bi-LSTM for objective (*iii*). For the former, there is an additional CRF layer, which helps improve the consistency of the output. *For subtask TC*, there are two distinct FFNs, feeding input representation, which are obtained in the same manner as for subtask SI. One of the two FFNs is for sentence representation, and the other one is for the representation of tokens in the propaganda span. The propaganda span representation is obtained by concatenating representation of the begin-of-sentence token, span start token, span end token, and aggregated representation by attention and max-pooling. *For both subtasks*, these architectures were trained independently with different BERT, GPT-2, XLNet, XLM, RoBERTa, or XLM-RoBERTa Transformers; and the resulting models were combined in ensembles.

Team Inno (Grigorev and Ivanov, 2020) (TC:7) used RoBERTa with cost-sensitive learning for subtask TC. They experimented with undersampling, i.e. removing examples from the bigger classes, as well as with modeling the context. They also tried various pre-trained Transformers, but obtained worse results.

Team JUST (Altiti et al., 2020) (TC:15) based its approach to the task on the BERT uncased pre-trained language model, which used 12 transformer layers that were trained for 15 epochs.

Team LTIatCMU (Khosla et al., 2020) (SI:4) used a multi-granular BERT BiLSTM for subtask SI. It used additional syntactic, semantic and pragmatic affect features at the word, sentence and document level. It was jointly trained on token and sentence propaganda classification, with class balancing. In addition, BERT was fine-tuned to persuasive language on about 10,000 articles from propaganda websites, which turned out to be important in their experiments.

Team newsSweeper (Singh et al., 2020) (SI: 13, TC: 5) used BERT with BIOE encoding for subtask SI. For the TC subtask, their official run used RoBERTa to obtain representations for the span and for the sentence, which they concatenated. The team further experimented (*i*) with other Transformers (BERT, RoBERTa, SpanBERT, and GPT-2), (*ii*) with other sequence labeling schemes (P/NP, BIO, BIOES), (*iii*) with concatenating different hidden layers of BERT to obtain a token representation, and (*iv*) with POS tags, as well as (*v*) with different neural architectures.

Team NLFIIT (Martinkovic et al., 2020) (SI: 17, TC: 16) used various combinations of neural architecture and embeddings and found out that ELMo combined with BiLSTM (and self attention for subtask TC) yielded the best performance.

Team NoPropaganda (Dimov et al., 2020) (SI: 7, TC: 6) used the LasetTagger model with the BERT-base encoder for subtask SI. R-BERT was used for subtask TC.

Team NTUAAILS (Arsenos and Siolas, 2020) (SI: 27, TC: 25) used a residual biLSTM fed with pre-trained ELMo embeddigns for subtask SI. A biLSTM was used for subtask TC as well, but this time fed with GloVe word embeddings

Team PsuedoProp (Chauhan and Diddee, 2020) (SI:14) focused on subtask SI. They pre-classified sentences as propaganda or not using an ensemble of XLNet and RoBERTa, before fine-tuning a BERT-based CRF sequence tagger to identify the exact spans.

Team SkoltechNLP (Dementieva et al., 2020) (SI: 25, TC: 26) fine-tuned BERT for SI, expanding the original training set through data augmentation techniques based on distributional semantics.

Team SocCogCom (Krishnamurthy et al., 2020) (TC:13) approached subtask TC using BERT/ALBERT together with (*i*) semantic-level emotional salience features from CrystalFeel (Gupta and Yang, 2018), and (*ii*) word-level psycholinguistic features from the LIWC lexicon (Pennebaker et al., 2015). They further modeled the context, i.e. three words before and after the target propaganda snippet.

Team Solomon (Raj et al., 2020) (TC:4) addressed subtask TC with a system that combines a transfer learning model based on fine-tuned RoBERTa (integrating fragment and context information), an ensemble of binary classifiers for the smaller classes and a novel system to specifically handle *Repetition*: they used dynamic least common sub-sequence to assess the similarity between the fragment and the context, and then the fragment was considered to be a repetition if the score was greater than a threshold heuristically set with respect to the length of the fragment.

Team syrapropa (Li and Xiao, 2020) (SI: 25, TC: 20) fine-tuned SpanBERT, a variant of BERT for span detection, on the context of spans in terms of the surrounding non-propaganda text for subtask SI. For subtask TC, they used a hybrid model that consists of several submodels, each specializing in some of the relations. These models include (*i*) BERT, (*ii*) BERT with cost adjustment to address class imbalance, and (*iii*) feature-rich logistic regression. The latter uses features such as length, TF.IDF-weighted words, repetitions, superlatives, and lists of fixed phrases targeting specific propaganda techniques. The output from the hybrid model was further post-processed using some syntactic rules based on part of speech.

Team Transformers (Verma et al., 2020) (SI: 9, TC: 29) explored a manifold of models to address the SI subtask. They considered residual biLSTMs fed with ELMo representations as well as different variations of BERT and RoBERTa with CNNs

Team TTUI (Kim and Bethard, 2020) (SI: 20, TC: 14) proposed an ensemble of fine-tuned BERT and RoBERTa models. They observed that feeding as input to the neural network a chunk of multiple, possibly overlapping sentences yielded the best performance. Moreover, for subtask SI, they applied a post-processing to remove gaps in the predictions between adjacent words. For subtask TC, they showed that modeling the context did not help in their experiments.

Team UAIC1860 (Ermurachi and Gifu, 2020) (SI: 28, TC: 26) used traditional text representation techniques: character n-grams, word2vec embeddings, and TF.IDF-weighted word-based features. For both subtasks, these features were used in a Random Forest classifier. Additional experiments with Naïve Bayes, Logistic Regression and SVMs yielded worse results.

Team UMSIForeseer Jiang et al. (2020) (TC:17) focused on subtask TC. They fine-tuned BERT on the labeled training spans, using a mix of oversampling and undersampling that is leveraged using a bagging ensemble learner.

Team UNTLing (Petee and Palmer, 2020) (TC:27) used a logistic regression classifier for subtask TC with a number of features, including bag-of-words, embeddings, NE and VAD lexicon features. Their analysis highlights that NE are useful for *Loaded Language* and *Flag Waving*. The VAD features were useful for emotion-related techniques such as *Appeal to fear/prejudice* and *Doubt*. They performed some experiments on the development set for subtask SI after the deadline. They used CRF with a number of features including PoS, syntactic dependency between the token and the previous/next word, BoW of preceding/following tokens, and the GloVe embedding of the token.

Team UPB (Paraschiv and Cercel, 2020) (SI: 5, TC: 19) used models based on BERT-base. Rather than just using the pre-trained models, they used masked language models to domain-adapt it with 9M-articles with fake, suspicious, and hyperpartisan news articles. They used the same domain-adapted model for both subtasks. They further used CRF for subtask SI, and a softmax for subtask TC.

Team UTMN (Mikhalkova et al., 2020) (SI:23) addressed subtask SI by representing the texts as a concatenation of tokens and context embeddings, together with sentiment intensity from VADER. They avoided deep learning architectures in order to produce a computationally affordable model, namely logistic regression.

Team WMD (Daval-Frerot and Yannick, 2020) (SI: 33, TC: 21) used an ensemble of BERT-based models, LSTMs, SVMs, gradient boosting, and random forest together with character and word-level embeddings. In addition, they used a number of techniques for data augmentation: back-translation, synonym replacement and TF.IDF replacement, i.e., replacing unimportant words, according to their TF.IDF score, with other unimportant words.

Team YNU-HPCC (Dao et al., 2020) (SI: 16, TC: 22) participated in both subtasks using GloVe and BERT embeddings in combination with LSTMs, BiLSTMs, and XGBoost.

Team YNUtaoxin (Tao and Zhou, 2020) (SI:11) used BERT, RoBERTa and XLNet on subtask SI focusing on determining the optimal input sentence length for the networks.

B Errata

After the shared task has ended, we found a bug in the code for our evaluation tools, which affected both subtasks. Overall, its impact was limited, and the ranking computed with the fixed code did not change substantially, in particular for the top-ranked teams. Tables 8 and 9 show the corrected scores on the test sets for subtasks SI and TC, respectively. Any reference to the task results should refer to these numbers.

	Test			
Team	**Rnk**	**F_1**	**P**	**R**
Hitachi	1	**51.74**	55.76	48.27
ApplicaAI	2	49.88	59.33	43.02
aschern	3	49.59	52.57	46.93
LTIatCMU	4	48.16	50.35	46.15
UPB	5	46.63	57.50	39.22
Fragarach	6	46.43	53.44	41.05
NoPropaganda	7	45.17	55.05	38.29
YNUtaoxin	8	43.80	54.60	36.57
Transformers	9	43.77	49.05	39.52
SWEAT	10	43.69	52.13	37.61
DREAM	11	43.60	53.67	36.71
CyberWallE	12	43.59	40.99	46.54
newsSweeper	13	42.20	45.30	39.49
PsuedoProp	14	41.81	41.24	42.41
Solomon	15	41.26	53.69	33.51
NLFIIT	16	41.10	50.17	34.81
TTUI	17	40.76	**66.37**	29.41
PALI	18	40.73	52.10	33.44
YNUHPCC	19	40.46	36.35	45.63
UESTCICSA	20	40.41	55.58	31.74
BPGC	21	38.89	48.50	32.45
DoNotDistribute	22	37.92	41.47	34.92
UTMNandOCAS	23	37.71	37.12	38.31
Entropy	24	37.31	40.82	34.35
syrapropa	25	36.92	49.22	29.53
SkoltechNLP	26	34.36	45.77	27.51
NTUAAILS	27	34.36	45.62	27.55
UAIC1860	28	32.67	23.86	51.78
CCNI	29	29.68	37.73	24.46
NCCU-SMRG	30	27.69	16.70	**80.94**
3218IR	31	23.28	21.95	24.79
WMD	32	20.51	45.44	13.24
LS	33	18.14	33.20	12.47
HunAlize	34	3.15	22.39	1.69
YOLO	35	0.74	18.32	0.38
Baseline	36	0.32	13.04	0.16

Table 8: **Subtask 1: Span Identification (SI) performance on the test set using the fixed scorer.** The highest score for each measure is highlighted.

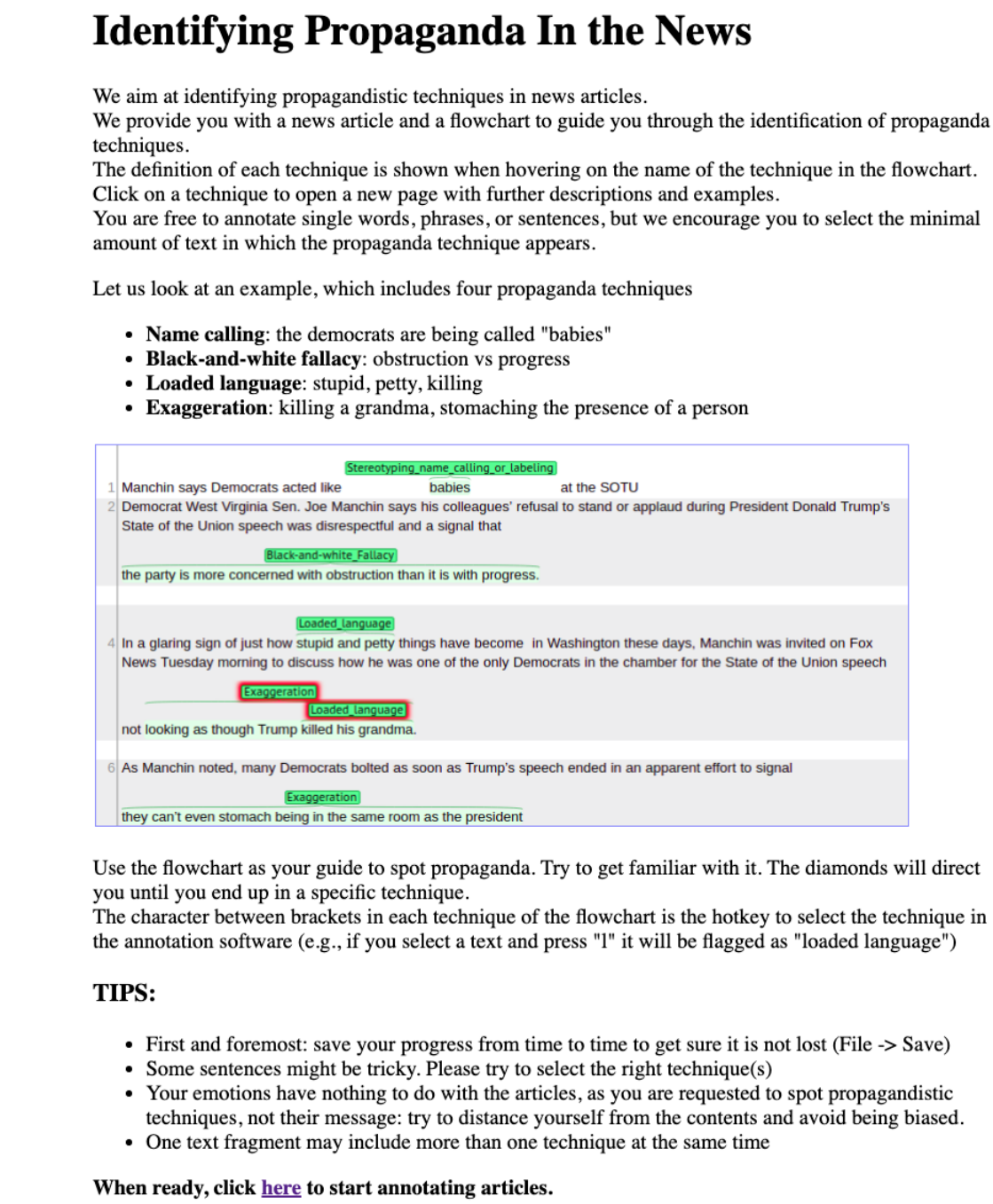
Rnk	Team	Overall	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	ApplicaAI	**63.74**	48.15	**47.06**	8.33	50.98	22.73	56.23	36.64	**70.47**	78.27	75.80	**58.79**	**58.97**	39.22	28.13
2	aschern	63.30	35.29	41.78	14.93	**53.57**	25.97	59.44	37.55	66.35	**78.32**	**76.68**	56.94	54.24	**42.55**	22.64
3	Hitachi	63.13	40.00	38.94	4.88	47.92	**36.62**	**63.43**	37.38	69.04	76.69	76.22	42.42	52.63	29.41	26.92
4	NoPropaganda	59.83	23.26	38.02	0.00	8.70	12.12	56.27	**39.43**	60.30	77.99	76.61	48.14	51.52	27.91	0.00
5	Solomon	59.39	26.67	39.39	4.88	42.00	19.61	61.82	32.00	61.84	75.30	70.75	42.53	50.75	38.10	0.00
6	CyberWallE	58.99	14.63	39.84	12.25	26.32	15.39	59.24	33.20	56.22	75.81	71.60	52.98	45.46	23.81	0.00
7	newsSweeper	58.44	41.67	42.80	13.04	25.00	19.72	56.55	37.10	60.53	75.32	74.23	20.69	50.75	21.62	8.00
8	Inno	57.99	43.14	45.09	10.71	40.40	24.32	58.65	35.39	59.41	73.31	74.30	24.89	43.75	29.63	19.36
9	djichen	57.71	37.50	34.58	20.83	43.81	22.79	60.00	32.18	56.19	73.99	70.43	36.36	30.77	27.91	18.87
10	PALI	57.43	27.59	36.59	**28.57**	39.62	30.59	58.97	28.57	61.62	74.29	69.43	25.44	39.22	40.82	20.90
11	DUTH	57.21	44.44	33.33	9.30	33.33	26.23	59.16	28.24	58.95	73.71	71.41	20.10	34.78	27.03	17.78
12	DiSaster	56.65	**51.16**	35.25	20.41	26.67	14.63	59.12	30.64	58.25	74.49	68.10	20.44	42.55	19.05	4.35
13	SocCogCom	55.81	28.57	31.62	9.76	26.51	6.35	60.40	34.86	54.26	72.18	67.34	18.88	40.91	23.53	10.00
14	TTUI	55.64	28.24	41.48	24.56	35.29	23.08	57.40	32.20	61.68	73.22	68.49	21.18	37.50	25.00	20.29
15	JUST	55.36	48.89	37.10	24.49	31.11	27.27	58.26	29.57	62.56	71.96	64.73	21.94	33.33	31.82	**28.57**
16	NLFIIT	55.25	29.63	24.89	9.30	32.10	0.00	55.66	30.30	63.32	72.55	69.30	21.55	41.67	13.64	0.00
17	UMSIForeseer	55.14	29.41	31.97	9.76	0.00	0.00	57.21	28.57	56.14	73.02	70.79	21.49	39.22	14.29	0.00
18	BPGC	54.81	30.77	33.65	12.50	40.00	24.18	53.78	33.47	58.93	71.58	67.51	23.74	40.00	20.69	20.90
19	UPB	54.30	20.00	30.00	0.00	19.72	16.95	52.55	30.62	55.87	70.09	68.86	20.00	34.62	22.86	4.88
20	syrapropa	54.25	31.58	29.77	19.61	30.61	21.51	56.14	28.10	57.02	71.47	68.44	30.77	29.03	28.57	9.09
21	WMD	52.35	7.14	29.20	5.00	0.00	5.71	54.42	26.36	52.08	69.76	64.67	14.82	6.90	7.41	0.00
22	YNUHPCC	50.50	29.79	26.40	13.79	38.38	3.45	51.04	21.54	55.56	68.08	62.33	17.72	27.59	17.78	15.00
23	UESTCICSA	49.94	12.77	22.75	0.00	11.27	3.74	44.99	25.44	53.14	68.23	66.88	27.96	41.38	28.57	3.70
24	DoNotDistribute	49.72	22.64	29.75	4.55	24.49	14.89	46.25	27.23	53.76	68.44	60.65	19.44	28.07	12.25	9.68
25	NTUAAILS	46.37	36.36	28.31	9.76	0.00	0.00	48.75	29.66	46.47	65.79	54.55	18.43	13.79	11.43	4.08
26	UAIC1860	41.17	17.39	22.12	9.76	2.90	7.60	36.41	21.01	38.78	62.33	42.97	11.16	11.43	5.56	4.26
27	UNTLing	39.11	4.35	5.29	0.00	0.00	2.86	32.65	11.82	40.48	62.57	36.74	7.78	17.65	0.00	0.00
28	HunAlize	37.10	6.45	11.83	0.00	8.00	7.84	31.76	23.81	29.95	58.59	15.82	2.09	4.55	0.00	0.00
29	Transformers	26.54	0.00	0.00	0.00	0.00	0.00	0.98	0.00	0.00	47.55	24.06	2.86	0.00	0.00	0.00
30	Baseline	25.20	0.00	3.68	0.00	0.00	11.56	29.14	14.42	6.20	46.48	0.00	19.26	0.00	0.00	0.00
31	IJSE8	20.62	0.00	7.07	0.00	0.00	2.27	17.07	10.70	8.57	39.32	15.39	4.92	0.00	0.00	0.00
32	Entropy	20.50	3.92	7.12	0.00	2.27	7.41	12.81	6.39	4.95	38.02	15.14	5.83	0.00	0.00	6.78

Table 9: **Technique classification F_1 performance on the test set using the fixed scorer**. The systems are ordered based on the final ranking. Columns **1** to **14** show the performance for each of the fourteen propaganda techniques (cf. Section 2). The best score for each technique is shown in bold.

C Annotation Instructions

Below, we show a series of snapshots of the actual annotation instructions and propaganda techniques definitions and examples that we showed to the human annotators. These are also available online:

- `http://propaganda.qcri.org/annotations/`
- `https://propaganda.qcri.org/annotations/definitions.html`



Identifying Propaganda In the News

We aim at identifying propagandistic techniques in news articles.
We provide you with a news article and a flowchart to guide you through the identification of propaganda techniques.
The definition of each technique is shown when hovering on the name of the technique in the flowchart.
Click on a technique to open a new page with further descriptions and examples.
You are free to annotate single words, phrases, or sentences, but we encourage you to select the minimal amount of text in which the propaganda technique appears.

Let us look at an example, which includes four propaganda techniques

- **Name calling**: the democrats are being called "babies"
- **Black-and-white fallacy**: obstruction vs progress
- **Loaded language**: stupid, petty, killing
- **Exaggeration**: killing a grandma, stomaching the presence of a person

Use the flowchart as your guide to spot propaganda. Try to get familiar with it. The diamonds will direct you until you end up in a specific technique.
The character between brackets in each technique of the flowchart is the hotkey to select the technique in the annotation software (e.g., if you select a text and press "l" it will be flagged as "loaded language")

TIPS:

- First and foremost: save your progress from time to time to get sure it is not lost (File -> Save)
- Some sentences might be tricky. Please try to select the right technique(s)
- Your emotions have nothing to do with the articles, as you are requested to spot propagandistic techniques, not their message: try to distance yourself from the contents and avoid being biased.
- One text fragment may include more than one technique at the same time

When ready, click here to start annotating articles.

Figure 7: Instruction for the annotators.

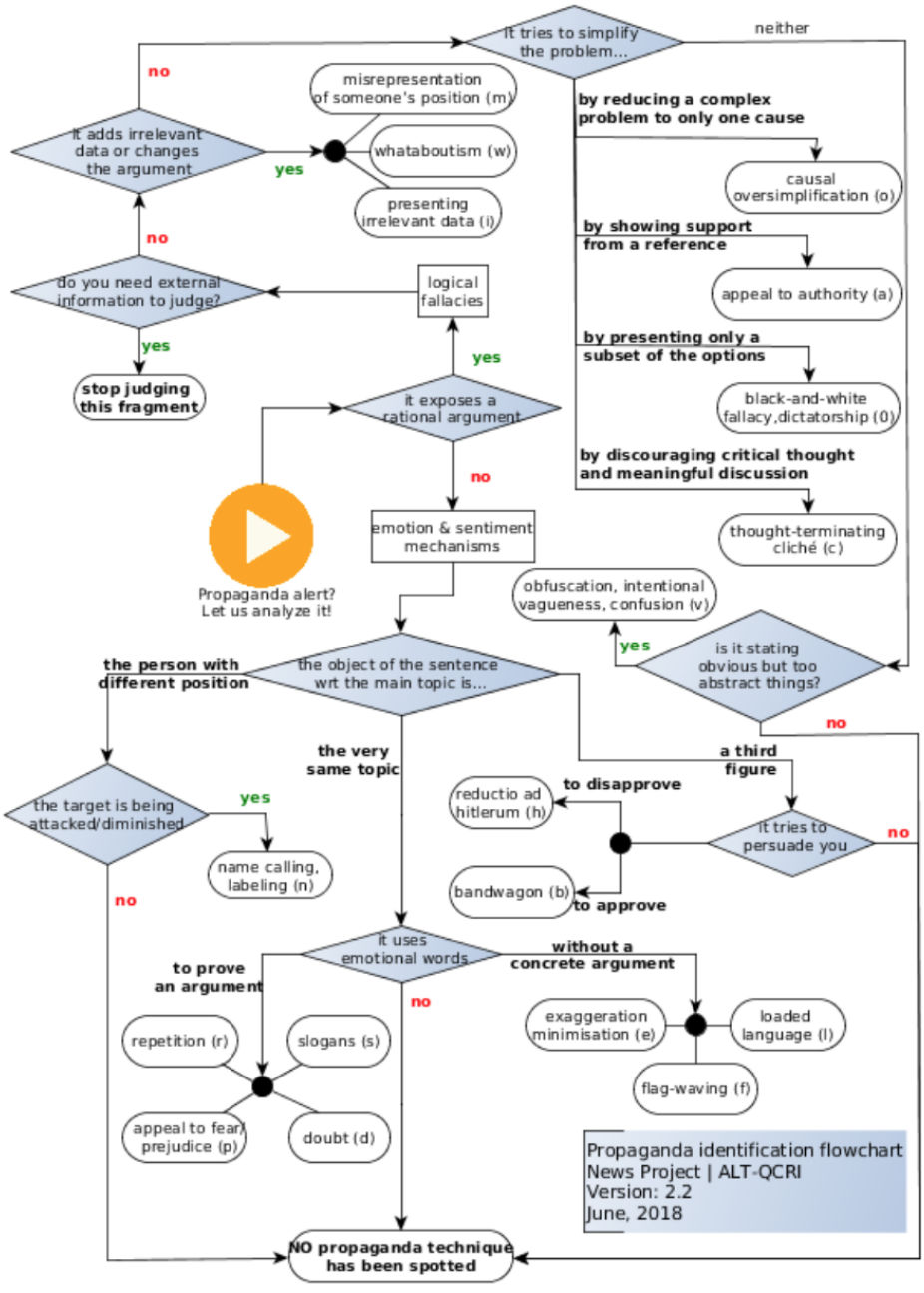


Figure 8: Annotation instructions: hierarchical diagram to guide the choice of technique.

Definitions

1. Presenting Irrelevant Data (Red Herring)

Introducing irrelevant material to the issue being discussed, so that everyone's attention is diverted away from the points made.

Example 1: In politics, defending one's own policies regarding public safety - "I have worked hard to help eliminate criminal activity. What we need is economic growth that can only come from the hands of leadership."
Example 2: "You may claim that the death penalty is an ineffective deterrent against crime -- but what about the victims of crime? How do you think surviving family members feel when they see the man who murdered their son kept in prison at their expense? Is it right that they should pay for their son's murderer to be fed and housed?"

2. Misrepresentation of Someone's Position (Straw Man)

When an opponent's proposition is substituted with a similar one which is then refuted in place of the original proposition.

Example: Zebedee: What is your view on the Christian God?
Mike: I don't believe in any gods, including the Christian one.
Zebedee: So you think that we are here by accident, and all this design in nature is pure chance, and the universe just created itself?
Mike: You got all that from me stating that I just don't believe in any gods?
Explanation: Mike made one claim: that he does not believe in any gods. From that, we can deduce a few things, like he is not a theist, he is not a practicing Christian, Catholic, Jew, or a member of any other religion that requires the belief in a god, but we cannot deduce that he believes we are all here by accident, nature is chance, and the universe created itself.

3. Whataboutism

A technique that attempts to discredit an opponent's position by charging them with hypocrisy without directly disproving their argument.

Example 1: a nation deflects criticism of its recent human rights violations by pointing to the history of slavery in the United States.
Example 2: "Qatar spending profusely on Neymar, not fighting terrorism"

4. Causal Oversimplification

Assuming a single cause or reason when there are actually multiple causes for an issue.
It includes transferring blame to one person or group of people without investigating the complexities of the issue

Example 1: "President Trump has been in office for a month and gas prices have been skyrocketing. The rise in gas prices is because of President Trump."
Example 2: The reason New Orleans was hit so hard with the hurricane was because of all the immoral people who live there.
Explanation: This was an actual argument seen in the months that followed hurricane Katrina. Ignoring the validity of the claims being made, the arguer is blaming a natural disaster on a group of people.
Example 3: if France had not have declared war on Germany then world war two would have never happened.

5. Obfuscation, Intentional vagueness, Confusion

Using words which are deliberately not clear so that the audience may have its own interpretations.
For example when an unclear phrase with multiple definitions is used within the argument and, therefore, it does not support the conclusion.

Example: It is a good idea to listen to victims of theft. Therefore if the victims say to have the thief shot, then you should do that.
Explanation: the definition for "listen to" is equivocated here. In the first case it means listen to their personal account of the experience of being a victim of theft. Empathize with them. In the second case "listen to" means carry out a punishment of their choice.

6. Appeal to authority

Stating that a claim is true simply because a valid authority or expert on the issue said it wa true, without any other supporting evidence offered. We consider the special case in which the reference is not an authority or an expert in this technique, altough it is referred to as Testimonial in literature.

Example: Richard Dawkins, an evolutionary biologist and perhaps the foremost expert in the field, says that evolution is true. Therefore, it's true.
Explanation: Richard Dawkins certainly knows about evolution, and he can confidently tell us that it is true, but that doesn't make it true. What makes it true is the preponderance of evidence for the theory.
Example 2: "According to Serena Williams, our foreign policy is the best on Earth. So we are in the right direction."
Details: since there is a chance that any authority can be wrong, it is reasonable to defer to an authority to support a claim, but the authority should not be the only justification to accept the claim, otherwise the Appeal-to-Authority fallacy is committed.

7. Black-and-white Fallacy, Dictatorship

Presenting two alternative options as the only possibilities, when in fact more possibilities exist. As an the extreme case, tell the audience exactly what actions to take, eliminating any other possible choices (Dictatorship).

Example 1: You must be a Republican or Democrat. You are not a Democrat. Therefore, you must be a Republican
Example 2: I thought you were a good person, but you weren't at church today.
Explanation: The assumption here is that if one doesn't attend chuch, one must be bad. Of course, good people exist who don't go to church, and good church-going people could have had a really good reason not to be in church.
Example 3: There is no alternative to war

8. Name calling or labeling

Labeling the object of the propaganda campaign as either something the target audience fears, hates, finds undesirable or loves, praises.

Examples: "Republican congressweasels", "Bush the Lesser" (note that lesser does not refer to "the second", but it is pejorative)

9. Loaded Language

Using specific words and phrases with strong emotional implications (either positive or negative) to influence an audience.

Example 1: "[...] a lone lawmaker's childish shouting. ".
Example 2: "how stupid and petty things have become in Washington"

10. Exaggeration or Minimisation

Either representing something in an excessive manner: making things larger, better, worse (e.g., "the best of the best", "quality guaranteed") or making something seem less important or smaller than it really is (e.g., saying that an insult was just a joke).

Example 1: "Democrats bolted as soon as Trump's speech ended in an apparent effort to signal they can't even stomach being in the same room as the president "
Example 2: "We're going to have unbelievable intelligence"
Example 3: I was not fighting with her; we were just playing.

11. Flag-waving

Playing on strong national feeling (or to any group; e.g., race, gender, political preference) to justify or promote an action or idea

Example 1: "patriotism mean no questions" (this is also a slogan)
Example 2: "entering this war will make us have a better future in our country."

12. Doubt

Questioning the credibility of someone or something.

Example: A candidate talks about his opponent and says: Is he ready to be the Mayor?

13. Appeal to fear/prejudice

Seeking to build support for an idea by instilling anxiety and/or panic in the population towards an alternative.
In some cases the support is built based on preconceived judgements.

Example 1: "either we go to war or we will perish" (this is also aBlack and White fallacy))
Example 2: "we must stop those refugees as they are terrorists"

14. Slogans

A brief and striking phrase that may include labeling and stereotyping. Slogans tend to act as emotional appeals.

Example 1: "The more women at war . . . the sooner we win."
Example 2: "Make America great again!"

15. Thought-terminating cliché

Words or phrases that discourage critical thought and meaningful discussion about a given topic. They are typically short, generic sentences that offer seemingly simple answers to complex questions or that distract attention away from other lines of thought.

Examples: It is what it is; It's just common sense; You gotta do what you gotta do; Nothing is permanent except change; Better late than never; Mind your own business; Nobody's perfect; It doesn't matter; You can't change human nature.

16. Bandwagon

Attempting to persuade the target audience to join in and take the course of action because "everyone else is taking the same action".

Example 1: Would you vote for Clinton as president? 57% say yes
Example 2: 90% of citizens support our initiative. You should.

17. Reductio ad hitlerum

Persuading an audience to disapprove an action or idea by suggesting that the idea is popular with groups hated in contempt by the target audience. It can refer to any person or concept with a negative connotation.

Example 1: "Do you know who else was doing that ? Hitler!"
Example 2: "Only one kind of person can think in that way: a communist."

18. Repetition

Repeating the same message over and over again so that the audience will eventually accept it.

Proceedings of the 40th Annual Meeting of the Association for
Computational Linguistics (ACL), Philadelphia, July 2002, pp. 311-318.

BLEU: a Method for Automatic Evaluation of Machine Translation

Kishore Papineni, Salim Roukos, Todd Ward, and **Wei-Jing Zhu**
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
{papineni,roukos,toddward,weijing}@us.ibm.com

Abstract

Human evaluations of machine translation are extensive but expensive. Human evaluations can take months to finish and involve human labor that can not be reused. We propose a method of automatic machine translation evaluation that is quick, inexpensive, and language-independent, that correlates highly with human evaluation, and that has little marginal cost per run. We present this method as an automated understudy to skilled human judges which substitutes for them when there is need for quick or frequent evaluations.[1]

1 Introduction

1.1 Rationale

Human evaluations of machine translation (MT) weigh many aspects of translation, including *adequacy*, *fidelity* , and *fluency* of the translation (Hovy, 1999; White and O'Connell, 1994). A comprehensive catalog of MT evaluation techniques and their rich literature is given by Reeder (2001). For the most part, these various human evaluation approaches are quite expensive (Hovy, 1999). Moreover, they can take *weeks* or *months* to finish. This is a big problem because developers of machine translation systems need to monitor the effect of *daily* changes to their systems in order to weed out bad ideas from good ideas. We believe that MT progress stems from evaluation and that there is a logjam of fruitful research ideas waiting to be released from the evaluation bottleneck. Developers would benefit from an inexpensive automatic evaluation that is quick, language-independent, and correlates highly with human evaluation. We propose such an evaluation method in this paper.

[1] So we call our method the bilingual evaluation understudy, BLEU.

1.2 Viewpoint

How does one measure translation performance? *The closer a machine translation is to a professional human translation, the better it is.* This is the central idea behind our proposal. To judge the quality of a machine translation, one measures its closeness to one or more reference human translations according to a numerical metric. Thus, our MT evaluation system requires two ingredients:

1. a numerical "translation closeness" metric
2. a corpus of good quality human reference translations

We fashion our closeness metric after the highly successful *word error rate* metric used by the speech recognition community, appropriately modified for multiple reference translations and allowing for legitimate differences in word choice and word order. The main idea is to use a weighted average of variable length phrase matches against the reference translations. This view gives rise to a family of metrics using various weighting schemes. We have selected a promising baseline metric from this family.

In Section 2, we describe the baseline metric in detail. In Section 3, we evaluate the performance of BLEU. In Section 4, we describe a human evaluation experiment. In Section 5, we compare our baseline metric performance with human evaluations.

2 The Baseline BLEU Metric

Typically, there are many "perfect" translations of a given source sentence. These translations may vary in word choice or in word order even when they use the same words. And yet humans can clearly distinguish a good translation from a bad one. For example, consider these two candidate translations of a Chinese source sentence:

Example 1.

Candidate 1: `It is a guide to action which ensures that the military always obeys the commands of the party.`

Candidate 2: `It is to insure the troops forever hearing the activity guidebook that party direct.`

Although they appear to be on the same subject, they differ markedly in quality. For comparison, we provide three reference human translations of the same sentence below.

Reference 1: `It is a guide to action that ensures that the military will forever heed Party commands.`

Reference 2: `It is the guiding principle which guarantees the military forces always being under the command of the Party.`

Reference 3: `It is the practical guide for the army always to heed the directions of the party.`

It is clear that the good translation, Candidate 1, shares many words and phrases with these three reference translations, while Candidate 2 does not. We will shortly quantify this notion of sharing in Section 2.1. But first observe that Candidate 1 shares `"It is a guide to action"` with Reference 1, `"which"` with Reference 2, `"ensures that the military"` with Reference 1, `"always"` with References 2 and 3, `"commands"` with Reference 1, and finally `"of the party"` with Reference 2 (all ignoring capitalization). In contrast, Candidate 2 exhibits far fewer matches, and their extent is less.

It is clear that a program can rank Candidate 1 higher than Candidate 2 simply by comparing *n*-gram matches between each candidate translation and the reference translations. Experiments over large collections of translations presented in Section 5 show that this ranking ability is a general phenomenon, and not an artifact of a few toy examples.

The primary programming task for a BLEU implementor is to compare *n*-grams of the candidate with the *n*-grams of the reference translation and count the number of matches. These matches are position-independent. The more the matches, the better the candidate translation is. For simplicity, we first focus on computing unigram matches.

2.1 Modified *n*-gram precision

The cornerstone of our metric is the familiar *precision* measure. To compute precision, one simply counts up the number of candidate translation words (unigrams) which occur in any reference translation and then divides by the total number of words in the candidate translation. Unfortunately, MT systems can overgenerate "reasonable" words, resulting in improbable, but high-precision, translations like that of example 2 below. Intuitively the problem is clear: a reference word should be considered exhausted after a matching candidate word is identified. We formalize this intuition as the *modified unigram precision*. To compute this, one first counts the maximum number of times a word occurs in any single reference translation. Next, one clips the total count of each candidate word by its maximum reference count,[2] adds these clipped counts up, and divides by the total (unclipped) number of candidate words.

Example 2.

Candidate: the the `the the the the the.`

Reference 1: The `cat is on` the `mat.`

Reference 2: `There is a cat on the mat.`

Modified Unigram Precision $= 2/7$.[3]

In Example 1, Candidate 1 achieves a modified unigram precision of $17/18$; whereas Candidate 2 achieves a modified unigram precision of $8/14$. Similarly, the modified unigram precision in Example 2 is $2/7$, even though its standard unigram precision is $7/7$.

[2] $Count_{clip} = \min(Count, Max\_Ref\_Count)$. In other words, one truncates each word's count, if necessary, to not exceed the largest count observed in any single reference for that word.

[3] As a guide to the eye, we have underlined the important words for computing modified precision.

Modified *n*-gram precision is computed similarly for any *n*: all candidate *n*-gram counts and their corresponding maximum reference counts are collected. The candidate counts are clipped by their corresponding reference maximum value, summed, and divided by the total number of candidate *n*-grams. In Example 1, Candidate 1 achieves a modified bigram precision of 10/17, whereas the lower quality Candidate 2 achieves a modified bigram precision of 1/13. In Example 2, the (implausible) candidate achieves a modified bigram precision of 0. This sort of modified *n*-gram precision scoring captures two aspects of translation: adequacy and fluency. A translation using the same words (1-grams) as in the references tends to satisfy *adequacy.* The longer *n*-gram matches account for *fluency.* [4]

2.1.1 Modified *n*-gram precision on blocks of text

How do we compute modified *n*-gram precision on a multi-sentence test set? Although one typically evaluates MT systems on a corpus of entire documents, our basic unit of evaluation is the sentence. A source sentence may translate to many target sentences, in which case we abuse terminology and refer to the corresponding target sentences as a "sentence." We first compute the *n*-gram matches sentence by sentence. Next, we add the clipped *n*-gram counts for all the candidate sentences and divide by the number of candidate *n*-grams in the test corpus to compute a modified precision score, p_n, for the entire test corpus.

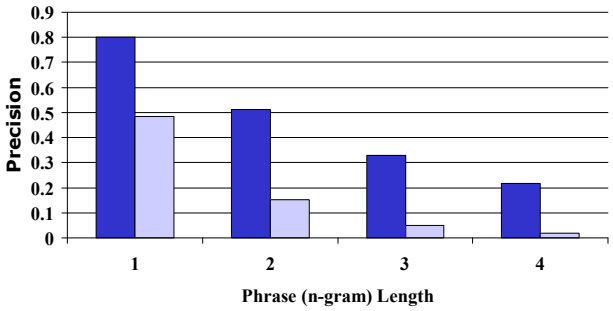
$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\sum_{C' \in \{Candidates\}} \sum_{n\text{-}gram' \in C'} Count(n\text{-}gram')}.$$

[4] BLEU only needs to match human judgment when averaged over a test corpus; scores on individual sentences will often vary from human judgments. For example, a system which produces the fluent phrase "East Asian economy" is penalized heavily on the longer *n*-gram precisions if all the references happen to read "economy of East Asia." The key to BLEU's success is that all systems are treated similarly and multiple human translators with different styles are used, so this effect cancels out in comparisons between systems.

2.1.2 Ranking systems using only modified *n*-gram precision

To verify that modified *n*-gram precision distinguishes between very good translations and bad translations, we computed the modified precision numbers on the output of a (good) human translator and a standard (poor) machine translation system using 4 reference translations for each of 127 source sentences. The average precision results are shown in Figure 1.

Figure 1: Distinguishing Human from Machine

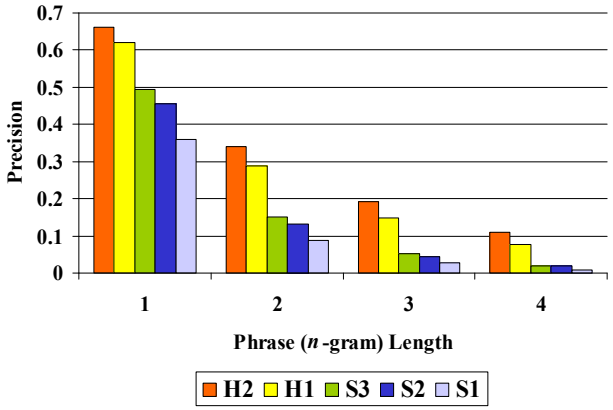


The strong signal differentiating human (high precision) from machine (low precision) is striking. The difference becomes stronger as we go from unigram precision to 4-gram precision. It appears that any single *n*-gram precision score can distinguish between a *good* translation and a *bad* translation. To be useful, however, the metric must also reliably distinguish between translations that do not differ so greatly in quality. Furthermore, it must distinguish between two human translations of differing quality. This latter requirement ensures the continued validity of the metric as MT approaches human translation quality.

To this end, we obtained a human translation by someone lacking native proficiency in both the source (Chinese) and the target language (English). For comparison, we acquired human translations of the same documents by a native English speaker. We also obtained machine translations by three commercial systems. These five "systems" — two humans and three machines — are scored against two reference professional human translations. The average modified *n*-gram precision results are shown in Figure 2.

Each of these *n*-gram statistics implies the same

Figure 2: Machine and Human Translations



ranking: H2 (Human-2) is better than H1 (Human-1), and there is a big drop in quality between H1 and S3 (Machine/System-3). S3 appears better than S2 which in turn appears better than S1. Remarkably, this is the *same rank order* assigned to these "systems" by human judges, as we discuss later. While there seems to be ample signal in any single *n*-gram precision, it is more robust to combine all these signals into a single number metric.

2.1.3 Combining the modified *n*-gram precisions

How should we combine the modified precisions for the various *n*-gram sizes? A weighted linear average of the modified precisions resulted in encouraging results for the 5 systems. However, as can be seen in Figure 2, the modified *n*-gram precision decays roughly exponentially with *n*: the modified unigram precision is much larger than the modified bigram precision which in turn is much bigger than the modified trigram precision. A reasonable averaging scheme must take this exponential decay into account; a weighted average of the logarithm of modified precisions satisifies this requirement.

BLEU uses the average logarithm with uniform weights, which is equivalent to using the geometric mean of the modified *n*-gram precisions.[5,6] Experimentally, we obtain the best correlation with monolingual human judgments using a maximum *n*-gram order of 4, although 3-grams and 5-grams give comparable results.

2.2 Sentence length

A candidate translation should be neither too long nor too short, and an evaluation metric should enforce this. To some extent, the *n*-gram precision already accomplishes this. *N*-gram precision penalizes spurious words in the candidate that do not appear in any of the reference translations. Additionally, modified precision is penalized if a word occurs more frequently in a candidate translation than its maximum reference count. This rewards using a word as many times as warranted and penalizes using a word more times than it occurs in any of the references. However, modified *n*-gram precision alone fails to enforce the proper translation length, as is illustrated in the short, absurd example below.

Example 3:

Candidate: `of the`

Reference 1: `It is a guide to action that ensures that the military will forever heed Party commands.`

Reference 2: `It is the guiding principle which guarantees the military forces always being under the command of the Party.`

Reference 3: `It is the practical guide for the army always to heed the directions of the party.`

Because this candidate is so short compared to the proper length, one expects to find inflated precisions: the modified unigram precision is 2/2, and the modified bigram precision is 1/1.

2.2.1 The trouble with recall

Traditionally, precision has been paired with recall to overcome such length-related problems. However, BLEU considers *multiple* reference translations, each of which may use a different word choice to translate the same source word. Furthermore, a good candidate translation will only use (recall) one of these possible choices, but not all. Indeed, recalling all choices leads to a bad translation. Here is an example.

[5]The geometric average is harsh if any of the modified precisions vanish, but this should be an extremely rare event in test corpora of reasonable size (for $N_{max} \leq 4$).

[6]Using the geometric average also yields slightly stronger correlation with human judgments than our best results using an arithmetic average.

Example 4:

Candidate 1: `I always invariably perpetually do.`

Candidate 2: `I always do.`

Reference 1: `I always do.`

Reference 2: `I invariably do.`

Reference 3: `I perpetually do.`

The first candidate recalls more words from the references, but is obviously a poorer translation than the second candidate. Thus, naïve recall computed over the set of all reference words is not a good measure. Admittedly, one could align the reference translations to discover synonymous words and compute recall on concepts rather than words. But, given that reference translations vary in length and differ in word order and syntax, such a computation is complicated.

2.2.2 Sentence brevity penalty

Candidate translations longer than their references are already penalized by the modified *n*-gram precision measure: there is no need to penalize them again. Consequently, we introduce a multiplicative *brevity penalty* factor. With this brevity penalty in place, a high-scoring candidate translation must now match the reference translations in length, in word choice, and in word order. Note that neither this brevity penalty nor the modified *n*-gram precision length effect directly considers the source length; instead, they consider the range of reference translation lengths in the target language.

We wish to make the brevity penalty 1.0 when the candidate's length is the same as any reference translation's length. For example, if there are three references with lengths 12, 15, and 17 words and the candidate translation is a terse 12 words, we want the brevity penalty to be 1. We call the closest reference sentence length the "*best match length.*"

One consideration remains: if we computed the brevity penalty sentence by sentence and averaged the penalties, then length deviations on short sentences would be punished harshly. Instead, we compute the brevity penalty over the entire corpus to allow some freedom at the sentence level. We first compute the test corpus' effective reference length, r, by summing the best match lengths for each candidate sentence in the corpus. We choose the brevity penalty to be a decaying exponential in r/c, where c is the total length of the candidate translation corpus.

2.3 BLEU details

We take the geometric mean of the test corpus' modified precision scores and then multiply the result by an exponential brevity penalty factor. Currently, case folding is the only text normalization performed before computing the precision.

We first compute the geometric average of the modified n-gram precisions, p_n, using n-grams up to length N and positive weights w_n summing to one.

Next, let c be the length of the candidate translation and r be the effective reference corpus length. We compute the brevity penalty BP,

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

Then,

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

The ranking behavior is more immediately apparent in the log domain,

$$\log \mathrm{BLEU} = \min(1 - \frac{r}{c}, 0) + \sum_{n=1}^{N} w_n \log p_n.$$

In our baseline, we use $N = 4$ and uniform weights $w_n = 1/N$.

3 The BLEU Evaluation

The BLEU metric ranges from 0 to 1. Few translations will attain a score of 1 unless they are identical to a reference translation. For this reason, even a human translator will not necessarily score 1. It is important to note that the more reference translations per sentence there are, the higher the score is. Thus, one must be cautious making even "rough" comparisons on evaluations with different numbers of reference translations: on a test corpus of about 500 sentences (40 general news stories), a human translator scored 0.3468 against four references and scored 0.2571 against two references. Table 1 shows the BLEU scores of the 5 systems against two references on this test corpus.

The MT systems S2 and S3 are very close in this metric. Hence, several questions arise:

Table 1: BLEU on 500 sentences

S1	S2	S3	H1	H2
0.0527	0.0829	0.0930	0.1934	0.2571

Table 2: Paired t-statistics on 20 blocks

	S1	S2	S3	H1	H2
Mean	0.051	0.081	0.090	0.192	0.256
StdDev	0.017	0.025	0.020	0.030	0.039
t	—	6	3.4	24	11

- Is the difference in BLEU metric reliable?
- What is the variance of the BLEU score?
- If we were to pick another random set of 500 sentences, would we still judge S3 to be better than S2?

To answer these questions, we divided the test corpus into 20 blocks of 25 sentences each, and computed the BLEU metric on these blocks individually. We thus have 20 samples of the BLEU metric for each system. We computed the means, variances, and paired t-statistics which are displayed in Table 2. The t-statistic compares each system with its left neighbor in the table. For example, $t = 6$ for the pair S1 and S2.

Note that the numbers in Table 1 are the BLEU metric on an aggregate of 500 sentences, but the means in Table 2 are averages of the BLEU metric on aggregates of 25 sentences. As expected, these two sets of results are close for each system and differ only by small finite block size effects. Since a paired t-statistic of 1.7 or above is 95% significant, the differences between the systems' scores are statistically very significant. The reported variance on 25-sentence blocks serves as an upper bound to the variance of sizeable test sets like the 500 sentence corpus.

How many reference translations do we need? We simulated a single-reference test corpus by randomly selecting one of the 4 reference translations as the single reference for each of the 40 stories. In this way, we ensured a degree of stylistic variation. The systems maintain the same rank order as with multiple references. This outcome suggests that we may use a big test corpus with a single reference translation, provided that the translations are not all from the same translator.

4 The Human Evaluation

We had two groups of human judges. The first group, called the monolingual group, consisted of 10 native speakers of English. The second group, called the bilingual group, consisted of 10 native speakers of Chinese who had lived in the United States for the past several years. None of the human judges was a professional translator. The humans judged our 5 standard systems on a Chinese sentence subset extracted at random from our 500 sentence test corpus. We paired each source sentence with each of its 5 translations, for a total of 250 pairs of Chinese source and English translations. We prepared a web page with these translation pairs randomly ordered to disperse the five translations of each source sentence. All judges used this same webpage and saw the sentence pairs in the same order. They rated each translation from 1 (very bad) to 5 (very good). The monolingual group made their judgments based only on the translations' readability and fluency.

As must be expected, some judges were more liberal than others. And some sentences were easier to translate than others. To account for the intrinsic difference between judges and the sentences, we compared each judge's rating for a sentence across systems. We performed four pairwise t-test comparisons between adjacent systems as ordered by their aggregate average score.

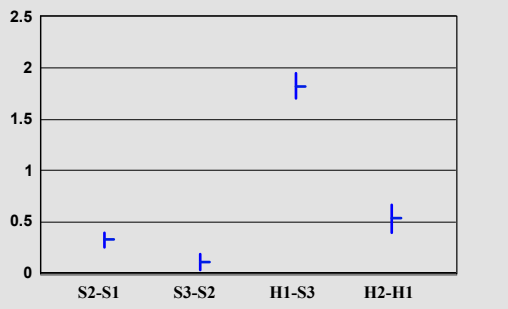
4.1 Monolingual group pairwise judgments

Figure 3 shows the mean difference between the scores of two consecutive systems and the 95% confidence interval about the mean. We see that S2 is quite a bit better than S1 (by a mean opinion score difference of 0.326 on the 5-point scale), while S3 is judged a little better (by 0.114). Both differences are significant at the 95% level.[7] The human H1 is much better than the best system, though a bit worse than human H2. This is not surprising given that H1 is not a native speaker of either Chinese or English,

[7] The 95% confidence interval comes from t-test, assuming that the data comes from a T-distribution with N degrees of freedom. N varied from 350 to 470 as some judges have skipped some sentences in their evaluation. Thus, the distribution is close to Gaussian.

whereas H2 is a native English speaker. Again, the difference between the human translators is significant beyond the 95% level.

Figure 3: Monolingual Judgments - pairwise differential comparison

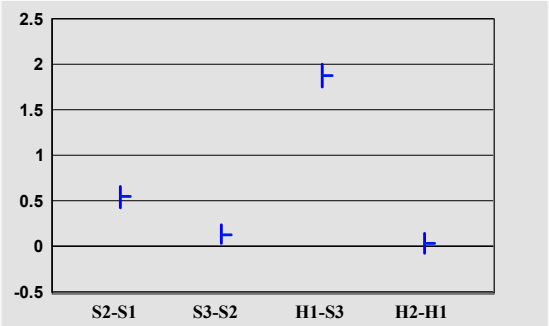


+95%	0.400	0.194	1.945	0.670
-95%	0.252	0.034	1.705	0.400
monolingual	0.326	0.114	1.825	0.535

4.2 Bilingual group pairwise judgments

Figure 4 shows the same results for the bilingual group. They also find that S3 is slightly better than S2 (at 95% confidence) though they judge that the human translations are much closer (indistinguishable at 95% confidence), suggesting that the bilinguals tended to focus more on adequacy than on fluency.

Figure 4: Bilingual Judgments - pairwise differential comparison



+95%	0.667	0.238	2.007	0.145
-95%	0.435	0.042	1.759	-0.069
bilingual	0.551	0.140	1.883	0.038

5 BLEU vs The Human Evaluation

Figure 5 shows a linear regression of the monolingual group scores as a function of the BLEU score over two reference translations for the 5 systems. The high correlation coefficient of 0.99 indicates that BLEU tracks human judgment well. Particularly interesting is how well BLEU distinguishes between S2 and S3 which are quite close. Figure 6 shows the comparable regression results for the bilingual group. The correlation coefficient is 0.96.

Figure 5: BLEU predicts Monolingual Judgments

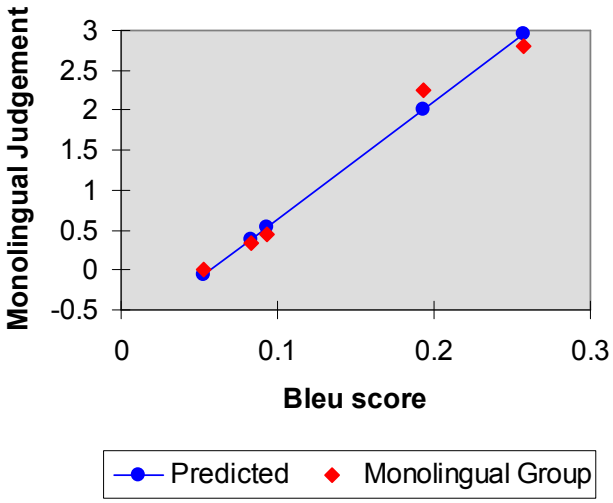
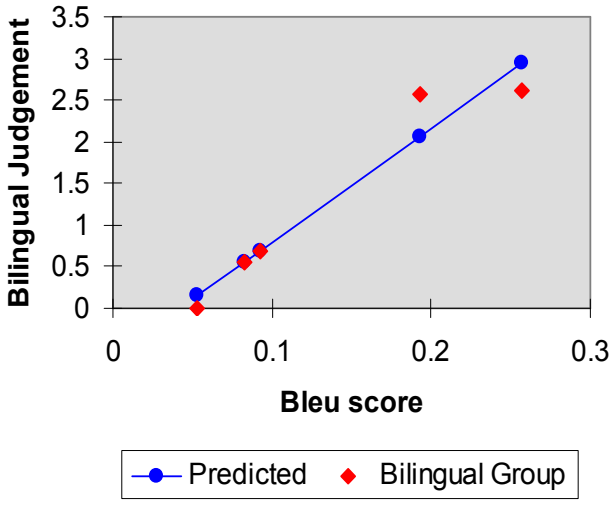


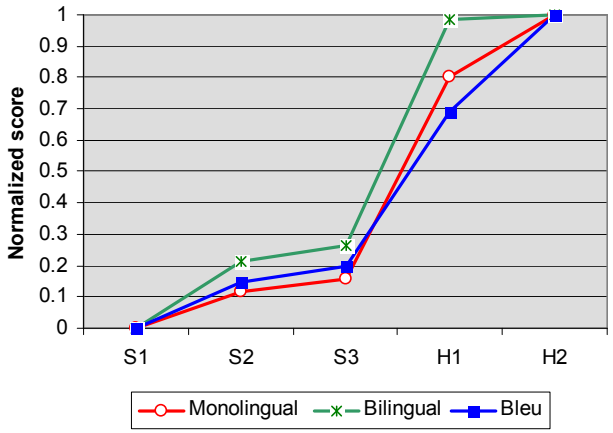
Figure 6: BLEU predicts Bilingual Judgments



We now take the worst system as a reference point and compare the BLEU scores with the human judg-

ment scores of the remaining systems relative to the worst system. We took the BLEU, monolingual group, and bilingual group scores for the 5 systems and linearly normalized them by their corresponding range (the maximum and minimum score across the 5 systems). The normalized scores are shown in Figure 7. This figure illustrates the high correlation between the BLEU score and the monolingual group. Of particular interest is the accuracy of BLEU's estimate of the small difference between S2 and S3 and the larger difference between S3 and H1. The figure also highlights the relatively large gap between MT systems and human translators.[8] In addition, we surmise that the bilingual group was very forgiving in judging H1 relative to H2 because the monolingual group found a rather large difference in the fluency of their translations.

Figure 7: BLEU vs Bilingual and Monolingual Judgments



6 Conclusion

We believe that BLEU will accelerate the MT R&D cycle by allowing researchers to rapidly home in on effective modeling ideas. Our belief is reinforced by a recent statistical analysis of BLEU's correlation with human judgment for translation into English from four quite different languages (Arabic, Chinese, French, Spanish) representing 3 different language families (Papineni et al., 2002)! BLEU's strength is that it correlates highly with human judgments by averaging out individual sentence judgment errors over a test corpus rather than attempting to divine the exact human judgment for every sentence: *quantity leads to quality.*

Finally, since MT and summarization can both be viewed as natural language generation from a textual context, we believe BLEU could be adapted to evaluating summarization or similar NLG tasks.

Acknowledgments This work was partially supported by the Defense Advanced Research Projects Agency and monitored by SPAWAR under contract No. N66001-99-2-8916. The views and findings contained in this material are those of the authors and do not necessarily reflect the position of policy of the Government and no official endorsement should be inferred.

We gratefully acknowledge comments about the geometric mean by John Makhoul of BBN and discussions with George Doddington of NIST. We especially wish to thank our colleagues who served in the monolingual and bilingual judge pools for their perseverance in judging the output of Chinese-English MT systems.

[8] Crossing this chasm for Chinese-English translation appears to be a significant challenge for the current state-of-the-art systems.

References

E.H. Hovy. 1999. Toward finely differentiated evaluation metrics for machine translation. In *Proceedings of the Eagles Workshop on Standards and Evaluation*, Pisa, Italy.

Kishore Papineni, Salim Roukos, Todd Ward, John Henderson, and Florence Reeder. 2002. Corpus-based comprehensive and diagnostic MT evaluation: Initial Arabic, Chinese, French, and Spanish results. In *Proceedings of Human Language Technology 2002*, San Diego, CA. To appear.

Florence Reeder. 2001. Additional mt-eval references. Technical report, International Standards for Language Engineering, Evaluation Working Group. http://issco-www.unige.ch/projects/isle/taxonomy2/

J.S. White and T. O'Connell. 1994. The ARPA MT evaluation methodologies: evolution, lessons, and future approaches. In *Proceedings of the First Conference of the Association for Machine Translation in the Americas*, pages 193–205, Columbia, Maryland.

Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

Nils Reimers and Iryna Gurevych
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

Abstract

BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) has set a new state-of-the-art performance on sentence-pair regression tasks like semantic textual similarity (STS). However, it requires that both sentences are fed into the network, which causes a massive computational overhead: Finding the most similar pair in a collection of 10,000 sentences requires about 50 million inference computations (~65 hours) with BERT. The construction of BERT makes it unsuitable for semantic similarity search as well as for unsupervised tasks like clustering.

In this publication, we present Sentence-BERT (SBERT), a modification of the pretrained BERT network that use siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity. This reduces the effort for finding the most similar pair from 65 hours with BERT / RoBERTa to about 5 seconds with SBERT, while maintaining the accuracy from BERT.

We evaluate SBERT and SRoBERTa on common STS tasks and transfer learning tasks, where it outperforms other state-of-the-art sentence embeddings methods.[1]

1 Introduction

In this publication, we present Sentence-BERT (SBERT), a modification of the BERT network using siamese and triplet networks that is able to derive semantically meaningful sentence embeddings[2]. This enables BERT to be used for certain new tasks, which up-to-now were not applicable for BERT. These tasks include large-scale semantic similarity comparison, clustering, and information retrieval via semantic search.

BERT set new state-of-the-art performance on various sentence classification and sentence-pair regression tasks. BERT uses a cross-encoder: Two sentences are passed to the transformer network and the target value is predicted. However, this setup is unsuitable for various pair regression tasks due to too many possible combinations. Finding in a collection of $n = 10\,000$ sentences the pair with the highest similarity requires with BERT $n \cdot (n-1)/2 = 49\,995\,000$ inference computations. On a modern V100 GPU, this requires about 65 hours. Similar, finding which of the over 40 million existent questions of Quora is the most similar for a new question could be modeled as a pair-wise comparison with BERT, however, answering a single query would require over 50 hours.

A common method to address clustering and semantic search is to map each sentence to a vector space such that semantically similar sentences are close. Researchers have started to input individual sentences into BERT and to derive fixed-size sentence embeddings. The most commonly used approach is to average the BERT output layer (known as BERT embeddings) or by using the output of the first token (the `[CLS]` token). As we will show, this common practice yields rather bad sentence embeddings, often worse than averaging GloVe embeddings (Pennington et al., 2014).

To alleviate this issue, we developed SBERT. The siamese network architecture enables that fixed-sized vectors for input sentences can be derived. Using a similarity measure like cosine-similarity or Manhatten / Euclidean distance, semantically similar sentences can be found. These similarity measures can be performed extremely efficient on modern hardware, allowing SBERT to be used for semantic similarity search as well as for clustering. The complexity for finding the

[1] Code available: https://github.com/UKPLab/sentence-transformers

[2] With *semantically meaningful* we mean that semantically similar sentences are close in vector space.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 3982–3992, Hong Kong, China, November 3–7, 2019.

most similar sentence pair in a collection of 10,000 sentences is reduced from 65 hours with BERT to the computation of 10,000 sentence embeddings (~5 seconds with SBERT) and computing cosine-similarity (~0.01 seconds). By using optimized index structures, finding the most similar Quora question can be reduced from 50 hours to a few milliseconds (Johnson et al., 2017).

We fine-tune SBERT on NLI data, which creates sentence embeddings that significantly outperform other state-of-the-art sentence embedding methods like InferSent (Conneau et al., 2017) and Universal Sentence Encoder (Cer et al., 2018). On seven Semantic Textual Similarity (STS) tasks, SBERT achieves an improvement of 11.7 points compared to InferSent and 5.5 points compared to Universal Sentence Encoder. On SentEval (Conneau and Kiela, 2018), an evaluation toolkit for sentence embeddings, we achieve an improvement of 2.1 and 2.6 points, respectively.

SBERT can be adapted to a specific task. It sets new state-of-the-art performance on a challenging argument similarity dataset (Misra et al., 2016) and on a triplet dataset to distinguish sentences from different sections of a Wikipedia article (Dor et al., 2018).

The paper is structured in the following way: Section 3 presents SBERT, section 4 evaluates SBERT on common STS tasks and on the challenging Argument Facet Similarity (AFS) corpus (Misra et al., 2016). Section 5 evaluates SBERT on SentEval. In section 6, we perform an ablation study to test some design aspect of SBERT. In section 7, we compare the computational efficiency of SBERT sentence embeddings in contrast to other state-of-the-art sentence embedding methods.

2 Related Work

We first introduce BERT, then, we discuss state-of-the-art sentence embedding methods.

BERT (Devlin et al., 2018) is a pre-trained transformer network (Vaswani et al., 2017), which set for various NLP tasks new state-of-the-art results, including question answering, sentence classification, and sentence-pair regression. The input for BERT for sentence-pair regression consists of the two sentences, separated by a special `[SEP]` token. Multi-head attention over 12 (base-model) or 24 layers (large-model) is applied and the output is passed to a simple regression function to derive the final label. Using this setup, BERT set a new state-of-the-art performance on the Semantic Textual Semilarity (STS) benchmark (Cer et al., 2017). RoBERTa (Liu et al., 2019) showed, that the performance of BERT can further improved by small adaptations to the pre-training process. We also tested XLNet (Yang et al., 2019), but it led in general to worse results than BERT.

A large disadvantage of the BERT network structure is that no independent sentence embeddings are computed, which makes it difficult to derive sentence embeddings from BERT. To bypass this limitations, researchers passed single sentences through BERT and then derive a fixed sized vector by either averaging the outputs (similar to average word embeddings) or by using the output of the special `CLS` token (for example: May et al. (2019); Zhang et al. (2019); Qiao et al. (2019)). These two options are also provided by the popular bert-as-a-service-repository[3]. Up to our knowledge, there is so far no evaluation if these methods lead to useful sentence embeddings.

Sentence embeddings are a well studied area with dozens of proposed methods. Skip-Thought (Kiros et al., 2015) trains an encoder-decoder architecture to predict the surrounding sentences. InferSent (Conneau et al., 2017) uses labeled data of the Stanford Natural Language Inference dataset (Bowman et al., 2015) and the Multi-Genre NLI dataset (Williams et al., 2018) to train a siamese BiLSTM network with max-pooling over the output. Conneau et al. showed, that InferSent consistently outperforms unsupervised methods like SkipThought. Universal Sentence Encoder (Cer et al., 2018) trains a transformer network and augments unsupervised learning with training on SNLI. Hill et al. (2016) showed, that the task on which sentence embeddings are trained significantly impacts their quality. Previous work (Conneau et al., 2017; Cer et al., 2018) found that the SNLI datasets are suitable for training sentence embeddings. Yang et al. (2018) presented a method to train on conversations from Reddit using siamese DAN and siamese transformer networks, which yielded good results on the STS benchmark dataset.

Humeau et al. (2019) addresses the run-time overhead of the cross-encoder from BERT and present a method (poly-encoders) to compute a score between m context vectors and pre-

[3] https://github.com/hanxiao/bert-as-service/

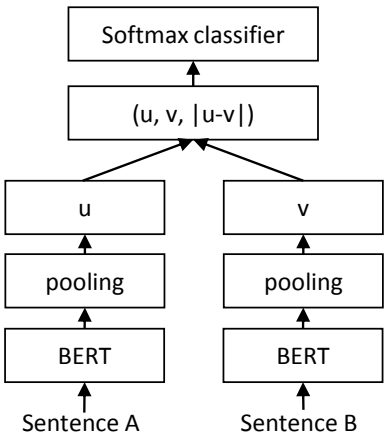


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

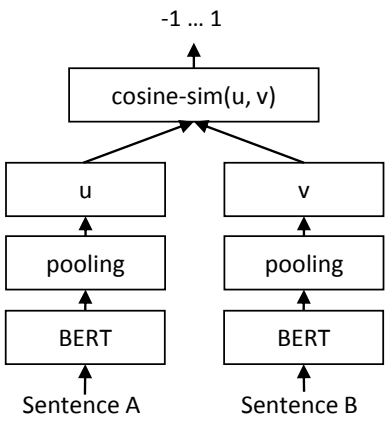


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

computed candidate embeddings using attention. This idea works for finding the highest scoring sentence in a larger collection. However, poly-encoders have the drawback that the score function is not symmetric and the computational overhead is too large for use-cases like clustering, which would require $O(n^2)$ score computations.

Previous neural sentence embedding methods started the training from a random initialization. In this publication, we use the pre-trained BERT and RoBERTa network and only fine-tune it to yield useful sentence embeddings. This reduces significantly the needed training time: SBERT can be tuned in less than 20 minutes, while yielding better results than comparable sentence embedding methods.

3 Model

SBERT adds a pooling operation to the output of BERT / RoBERTa to derive a fixed sized sentence embedding. We experiment with three pooling strategies: Using the output of the `CLS`-token, computing the mean of all output vectors (`MEAN`-strategy), and computing a max-over-time of the output vectors (`MAX`-strategy). The default configuration is `MEAN`.

In order to fine-tune BERT / RoBERTa, we create siamese and triplet networks (Schroff et al., 2015) to update the weights such that the produced sentence embeddings are semantically meaningful and can be compared with cosine-similarity.

The network structure depends on the available training data. We experiment with the following structures and objective functions.

Classification Objective Function. We concatenate the sentence embeddings u and v with the element-wise difference $|u-v|$ and multiply it with the trainable weight $W_t \in \mathbb{R}^{3n \times k}$:

$$o = \text{softmax}(W_t(u, v, |u - v|))$$

where n is the dimension of the sentence embeddings and k the number of labels. We optimize cross-entropy loss. This structure is depicted in Figure 1.

Regression Objective Function. The cosine-similarity between the two sentence embeddings u and v is computed (Figure 2). We use mean-squared-error loss as the objective function.

Triplet Objective Function. Given an anchor sentence a, a positive sentence p, and a negative sentence n, triplet loss tunes the network such that the distance between a and p is smaller than the distance between a and n. Mathematically, we minimize the following loss function:

$$max(||s_a - s_p|| - ||s_a - s_n|| + \epsilon, 0)$$

with s_x the sentence embedding for $a/n/p$, $||\cdot||$ a distance metric and margin ϵ. Margin ϵ ensures that s_p is at least ϵ closer to s_a than s_n. As metric we use Euclidean distance and we set $\epsilon = 1$ in our experiments.

3.1 Training Details

We train SBERT on the combination of the SNLI (Bowman et al., 2015) and the Multi-Genre NLI

Model	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
Avg. GloVe embeddings	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
Avg. BERT embeddings	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT CLS-vector	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
InferSent - Glove	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
Universal Sentence Encoder	64.49	67.80	64.61	76.83	73.18	74.92	**76.69**	71.22
SBERT-NLI-base	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
SBERT-NLI-large	72.27	**78.46**	**74.90**	80.99	76.25	**79.23**	73.75	76.55
SRoBERTa-NLI-base	71.54	72.49	70.80	78.74	73.69	77.77	74.46	74.21
SRoBERTa-NLI-large	**74.53**	77.00	73.18	**81.85**	**76.82**	79.10	74.29	**76.68**

Table 1: Spearman rank correlation ρ between the cosine similarity of sentence representations and the gold labels for various Textual Similarity (STS) tasks. Performance is reported by convention as $\rho \times 100$. STS12-STS16: SemEval 2012-2016, STSb: STSbenchmark, SICK-R: SICK relatedness dataset.

(Williams et al., 2018) dataset. The SNLI is a collection of 570,000 sentence pairs annotated with the labels *contradiction*, *eintailment*, and *neutral*. MultiNLI contains 430,000 sentence pairs and covers a range of genres of spoken and written text. We fine-tune SBERT with a 3-way softmax-classifier objective function for one epoch. We used a batch-size of 16, Adam optimizer with learning rate $2e{-}5$, and a linear learning rate warm-up over 10% of the training data. Our default pooling strategy is `MEAN`.

4 Evaluation - Semantic Textual Similarity

We evaluate the performance of SBERT for common Semantic Textual Similarity (STS) tasks. State-of-the-art methods often learn a (complex) regression function that maps sentence embeddings to a similarity score. However, these regression functions work pair-wise and due to the combinatorial explosion those are often not scalable if the collection of sentences reaches a certain size. Instead, we always use cosine-similarity to compare the similarity between two sentence embeddings. We ran our experiments also with negative Manhatten and negative Euclidean distances as similarity measures, but the results for all approaches remained roughly the same.

4.1 Unsupervised STS

We evaluate the performance of SBERT for STS without using any STS specific training data. We use the STS tasks 2012 - 2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016), the STS benchmark (Cer et al., 2017), and the SICK-Relatedness dataset (Marelli et al., 2014). These datasets provide labels between 0 and 5 on the semantic relatedness of sentence pairs. We showed in (Reimers et al., 2016) that Pearson correlation is badly suited for STS. Instead, we compute the Spearman's rank correlation between the cosine-similarity of the sentence embeddings and the gold labels. The setup for the other sentence embedding methods is equivalent, the similarity is computed by cosine-similarity. The results are depicted in Table 1.

The results shows that directly using the output of BERT leads to rather poor performances. Averaging the BERT embeddings achieves an average correlation of only 54.81, and using the `CLS`-token output only achieves an average correlation of 29.19. Both are worse than computing average GloVe embeddings.

Using the described siamese network structure and fine-tuning mechanism substantially improves the correlation, outperforming both InferSent and Universal Sentence Encoder substantially. The only dataset where SBERT performs worse than Universal Sentence Encoder is SICK-R. Universal Sentence Encoder was trained on various datasets, including news, question-answer pages and discussion forums, which appears to be more suitable to the data of SICK-R. In contrast, SBERT was pre-trained only on Wikipedia (via BERT) and on NLI data.

While RoBERTa was able to improve the performance for several supervised tasks, we only observe minor difference between SBERT and SRoBERTa for generating sentence embeddings.

4.2 Supervised STS

The STS benchmark (STSb) (Cer et al., 2017) provides is a popular dataset to evaluate supervised STS systems. The data includes 8,628 sentence pairs from the three categories *captions*, *news*, and *forums*. It is divided into train (5,749), dev (1,500) and test (1,379). BERT set a new state-of-the-art performance on this dataset by passing both sentences to the network and using a simple regres-

sion method for the output.

Model	Spearman
Not trained for STS	
Avg. GloVe embeddings	58.02
Avg. BERT embeddings	46.35
InferSent - GloVe	68.03
Universal Sentence Encoder	74.92
SBERT-NLI-base	77.03
SBERT-NLI-large	79.23
Trained on STS benchmark dataset	
BERT-STSb-base	84.30 ± 0.76
SBERT-STSb-base	84.67 ± 0.19
SRoBERTa-STSb-base	**84.92** ± 0.34
BERT-STSb-large	**85.64** ± 0.81
SBERT-STSb-large	84.45 ± 0.43
SRoBERTa-STSb-large	85.02 ± 0.76
Trained on NLI data + STS benchmark data	
BERT-NLI-STSb-base	**88.33** ± 0.19
SBERT-NLI-STSb-base	85.35 ± 0.17
SRoBERTa-NLI-STSb-base	84.79 ± 0.38
BERT-NLI-STSb-large	**88.77** ± 0.46
SBERT-NLI-STSb-large	86.10 ± 0.13
SRoBERTa-NLI-STSb-large	86.15 ± 0.35

Table 2: Evaluation on the STS benchmark test set. BERT systems were trained with 10 random seeds and 4 epochs. SBERT was fine-tuned on the STSb dataset, SBERT-NLI was pretrained on the NLI datasets, then fine-tuned on the STSb dataset.

We use the training set to fine-tune SBERT using the regression objective function. At prediction time, we compute the cosine-similarity between the sentence embeddings. All systems are trained with 10 random seeds to counter variances (Reimers and Gurevych, 2018).

The results are depicted in Table 2. We experimented with two setups: Only training on STSb, and first training on NLI, then training on STSb. We observe that the later strategy leads to a slight improvement of 1-2 points. This two-step approach had an especially large impact for the BERT cross-encoder, which improved the performance by 3-4 points. We do not observe a significant difference between BERT and RoBERTa.

4.3 Argument Facet Similarity

We evaluate SBERT on the Argument Facet Similarity (AFS) corpus by Misra et al. (2016). The AFS corpus annotated 6,000 sentential argument pairs from social media dialogs on three controversial topics: *gun control*, *gay marriage*, and *death penalty*. The data was annotated on a scale from 0 ("different topic") to 5 ("completely equivalent"). The similarity notion in the AFS corpus is fairly different to the similarity notion in the STS datasets from SemEval. STS data is usually descriptive, while AFS data are argumentative excerpts from dialogs. To be considered similar, arguments must not only make similar claims, but also provide a similar reasoning. Further, the lexical gap between the sentences in AFS is much larger. Hence, simple unsupervised methods as well as state-of-the-art STS systems perform badly on this dataset (Reimers et al., 2019).

We evaluate SBERT on this dataset in two scenarios: 1) As proposed by Misra et al., we evaluate SBERT using 10-fold cross-validation. A drawback of this evaluation setup is that it is not clear how well approaches generalize to different topics. Hence, 2) we evaluate SBERT in a cross-topic setup. Two topics serve for training and the approach is evaluated on the left-out topic. We repeat this for all three topics and average the results.

SBERT is fine-tuned using the Regression Objective Function. The similarity score is computed using cosine-similarity based on the sentence embeddings. We also provide the Pearson correlation r to make the results comparable to Misra et al. However, we showed (Reimers et al., 2016) that Pearson correlation has some serious drawbacks and should be avoided for comparing STS systems. The results are depicted in Table 3.

Unsupervised methods like tf-idf, average GloVe embeddings or InferSent perform rather badly on this dataset with low scores. Training SBERT in the 10-fold cross-validation setup gives a performance that is nearly on-par with BERT.

However, in the cross-topic evaluation, we observe a performance drop of SBERT by about 7 points Spearman correlation. To be considered similar, arguments should address the same claims and provide the same reasoning. BERT is able to use attention to compare directly both sentences (e.g. word-by-word comparison), while SBERT must map individual sentences from an unseen topic to a vector space such that arguments with similar claims and reasons are close. This is a much more challenging task, which appears to require more than just two topics for training to work on-par with BERT.

4.4 Wikipedia Sections Distinction

Dor et al. (2018) use Wikipedia to create a thematically fine-grained train, dev and test set for sentence embeddings methods. Wikipedia articles are separated into distinct sections focusing on certain aspects. Dor et al. assume that sen-

Model	r	ρ
Unsupervised methods		
tf-idf	46.77	42.95
Avg. GloVe embeddings	32.40	34.00
InferSent - GloVe	27.08	26.63
10-fold Cross-Validation		
SVR (Misra et al., 2016)	63.33	-
BERT-AFS-base	77.20	74.84
SBERT-AFS-base	76.57	74.13
BERT-AFS-large	78.68	76.38
SBERT-AFS-large	77.85	75.93
Cross-Topic Evaluation		
BERT-AFS-base	58.49	57.23
SBERT-AFS-base	52.34	50.65
BERT-AFS-large	62.02	60.34
SBERT-AFS-large	53.82	53.10

Table 3: Average Pearson correlation r and average Spearman's rank correlation ρ on the Argument Facet Similarity (AFS) corpus (Misra et al., 2016). Misra et al. proposes 10-fold cross-validation. We additionally evaluate in a cross-topic scenario: Methods are trained on two topics, and are evaluated on the third topic.

tences in the same section are thematically closer than sentences in different sections. They use this to create a large dataset of weakly labeled sentence triplets: The anchor and the positive example come from the same section, while the negative example comes from a different section of the same article. For example, from the Alice Arnold article: Anchor: *Arnold joined the BBC Radio Drama Company in 1988.*, positive: *Arnold gained media attention in May 2012.*, negative: *Balding and Arnold are keen amateur golfers.*

We use the dataset from Dor et al. We use the Triplet Objective, train SBERT for one epoch on the about 1.8 Million training triplets and evaluate it on the 222,957 test triplets. Test triplets are from a distinct set of Wikipedia articles. As evaluation metric, we use accuracy: Is the positive example closer to the anchor than the negative example?

Results are presented in Table 4. Dor et al. fine-tuned a BiLSTM architecture with triplet loss to derive sentence embeddings for this dataset. As the table shows, SBERT clearly outperforms the BiLSTM approach by Dor et al.

5 Evaluation - SentEval

SentEval (Conneau and Kiela, 2018) is a popular toolkit to evaluate the quality of sentence embeddings. Sentence embeddings are used as features for a logistic regression classifier. The logistic regression classifier is trained on various tasks in a 10-fold cross-validation setup and the prediction accuracy is computed for the test-fold.

Model	Accuracy
mean-vectors	0.65
skip-thoughts-CS	0.62
Dor et al.	0.74
SBERT-WikiSec-base	0.8042
SBERT-WikiSec-large	**0.8078**
SRoBERTa-WikiSec-base	0.7945
SRoBERTa-WikiSec-large	0.7973

Table 4: Evaluation on the Wikipedia section triplets dataset (Dor et al., 2018). SBERT trained with triplet loss for one epoch.

The purpose of SBERT sentence embeddings are not to be used for transfer learning for other tasks. Here, we think fine-tuning BERT as described by Devlin et al. (2018) for new tasks is the more suitable method, as it updates all layers of the BERT network. However, SentEval can still give an impression on the quality of our sentence embeddings for various tasks.

We compare the SBERT sentence embeddings to other sentence embeddings methods on the following seven SentEval transfer tasks:

- **MR**: Sentiment prediction for movie reviews snippets on a five start scale (Pang and Lee, 2005).
- **CR**: Sentiment prediction of customer product reviews (Hu and Liu, 2004).
- **SUBJ**: Subjectivity prediction of sentences from movie reviews and plot summaries (Pang and Lee, 2004).
- **MPQA**: Phrase level opinion polarity classification from newswire (Wiebe et al., 2005).
- **SST**: Stanford Sentiment Treebank with binary labels (Socher et al., 2013).
- **TREC**: Fine grained question-type classification from TREC (Li and Roth, 2002).
- **MRPC**: Microsoft Research Paraphrase Corpus from parallel news sources (Dolan et al., 2004).

The results can be found in Table 5. SBERT is able to achieve the best performance in 5 out of 7 tasks. The average performance increases by about 2 percentage points compared to InferSent as well as the Universal Sentence Encoder. Even though transfer learning is not the purpose of SBERT, it outperforms other state-of-the-art sentence embeddings methods on this task.

Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	Avg.
Avg. GloVe embeddings	77.25	78.30	91.17	87.85	80.18	83.0	72.87	81.52
Avg. fast-text embeddings	77.96	79.23	91.68	87.81	82.15	83.6	74.49	82.42
Avg. BERT embeddings	78.66	86.25	94.37	88.66	84.40	92.8	69.45	84.94
BERT CLS-vector	78.68	84.85	94.21	88.23	84.13	91.4	71.13	84.66
InferSent - GloVe	81.57	86.54	92.50	**90.38**	84.18	88.2	75.77	85.59
Universal Sentence Encoder	80.09	85.19	93.98	86.70	86.38	**93.2**	70.14	85.10
SBERT-NLI-base	83.64	89.43	94.39	89.86	88.96	89.6	**76.00**	87.41
SBERT-NLI-large	**84.88**	**90.07**	**94.52**	90.33	**90.66**	87.4	75.94	**87.69**

Table 5: Evaluation of SBERT sentence embeddings using the SentEval toolkit. SentEval evaluates sentence embeddings on different sentence classification tasks by training a logistic regression classifier using the sentence embeddings as features. Scores are based on a 10-fold cross-validation.

It appears that the sentence embeddings from SBERT capture well sentiment information: We observe large improvements for all sentiment tasks (MR, CR, and SST) from SentEval in comparison to InferSent and Universal Sentence Encoder.

The only dataset where SBERT is significantly worse than Universal Sentence Encoder is the TREC dataset. Universal Sentence Encoder was pre-trained on question-answering data, which appears to be beneficial for the question-type classification task of the TREC dataset.

Average BERT embeddings or using the CLS-token output from a BERT network achieved bad results for various STS tasks (Table 1), worse than average GloVe embeddings. However, for SentEval, average BERT embeddings and the BERT CLS-token output achieves decent results (Table 5), outperforming average GloVe embeddings. The reason for this are the different setups. For the STS tasks, we used cosine-similarity to estimate the similarities between sentence embeddings. Cosine-similarity treats all dimensions equally. In contrast, SentEval fits a logistic regression classifier to the sentence embeddings. This allows that certain dimensions can have higher or lower impact on the classification result.

We conclude that average BERT embeddings / CLS-token output from BERT return sentence embeddings that are infeasible to be used with cosine-similarity or with Manhatten / Euclidean distance. For transfer learning, they yield slightly worse results than InferSent or Universal Sentence Encoder. However, using the described fine-tuning setup with a siamese network structure on NLI datasets yields sentence embeddings that achieve a new state-of-the-art for the SentEval toolkit.

6 Ablation Study

We have demonstrated strong empirical results for the quality of SBERT sentence embeddings. In this section, we perform an ablation study of different aspects of SBERT in order to get a better understanding of their relative importance.

We evaluated different pooling strategies (MEAN, MAX, and CLS). For the classification objective function, we evaluate different concatenation methods. For each possible configuration, we train SBERT with 10 different random seeds and average the performances.

The objective function (classification vs. regression) depends on the annotated dataset. For the classification objective function, we train SBERT-base on the SNLI and the Multi-NLI dataset. For the regression objective function, we train on the training set of the STS benchmark dataset. Performances are measured on the development split of the STS benchmark dataset. Results are shown in Table 6.

	NLI	STSb
Pooling Strategy		
MEAN	**80.78**	**87.44**
MAX	79.07	69.92
CLS	79.80	86.62
Concatenation		
(u, v)	66.04	-
$(\|u - v\|)$	69.78	-
$(u * v)$	70.54	-
$(\|u - v\|, u * v)$	78.37	-
$(u, v, u * v)$	77.44	-
$(u, v, \|u - v\|)$	**80.78**	-
$(u, v, \|u - v\|, u * v)$	80.44	-

Table 6: SBERT trained on NLI data with the classification objective function, on the STS benchmark (STSb) with the regression objective function. Configurations are evaluated on the development set of the STSb using cosine-similarity and Spearman's rank correlation. For the concatenation methods, we only report scores with MEAN pooling strategy.

When trained with the classification objective function on NLI data, the pooling strategy has a rather minor impact. The impact of the concatenation mode is much larger. InferSent (Conneau

et al., 2017) and Universal Sentence Encoder (Cer et al., 2018) both use $(u, v, |u - v|, u * v)$ as input for a softmax classifier. However, in our architecture, adding the element-wise $u * v$ decreased the performance.

The most important component is the element-wise difference $|u - v|$. Note, that the concatenation mode is only relevant for training the softmax classifier. At inference, when predicting similarities for the STS benchmark dataset, only the sentence embeddings u and v are used in combination with cosine-similarity. The element-wise difference measures the distance between the dimensions of the two sentence embeddings, ensuring that similar pairs are closer and dissimilar pairs are further apart.

When trained with the regression objective function, we observe that the pooling strategy has a large impact. There, the `MAX` strategy perform significantly worse than `MEAN` or `CLS`-token strategy. This is in contrast to (Conneau et al., 2017), who found it beneficial for the BiLSTM-layer of InferSent to use `MAX` instead of `MEAN` pooling.

7 Computational Efficiency

Sentence embeddings need potentially be computed for Millions of sentences, hence, a high computation speed is desired. In this section, we compare SBERT to average GloVe embeddings, InferSent (Conneau et al., 2017), and Universal Sentence Encoder (Cer et al., 2018).

For our comparison we use the sentences from the STS benchmark (Cer et al., 2017). We compute average GloVe embeddings using a simple for-loop with python dictionary lookups and NumPy. InferSent[4] is based on PyTorch. For Universal Sentence Encoder, we use the TensorFlow Hub version[5], which is based on TensorFlow. SBERT is based on PyTorch. For improved computation of sentence embeddings, we implemented a smart batching strategy: Sentences with similar lengths are grouped together and are only padded to the longest element in a mini-batch. This drastically reduces computational overhead from padding tokens.

Performances were measured on a server with Intel i7-5820K CPU @ 3.30GHz, Nvidia Tesla V100 GPU, CUDA 9.2 and cuDNN. The results are depicted in Table 7.

Model	CPU	GPU
Avg. GloVe embeddings	6469	-
InferSent	137	1876
Universal Sentence Encoder	67	1318
SBERT-base	44	1378
SBERT-base - smart batching	83	2042

Table 7: Computation speed (sentences per second) of sentence embedding methods. Higher is better.

On CPU, InferSent is about 65% faster than SBERT. This is due to the much simpler network architecture. InferSent uses a single BiLSTM layer, while BERT uses 12 stacked transformer layers. However, an advantage of transformer networks is the computational efficiency on GPUs. There, SBERT with smart batching is about 9% faster than InferSent and about 55% faster than Universal Sentence Encoder. Smart batching achieves a speed-up of 89% on CPU and 48% on GPU. Average GloVe embeddings is obviously by a large margin the fastest method to compute sentence embeddings.

8 Conclusion

We showed that BERT out-of-the-box maps sentences to a vector space that is rather unsuitable to be used with common similarity measures like cosine-similarity. The performance for seven STS tasks was below the performance of average GloVe embeddings.

To overcome this shortcoming, we presented Sentence-BERT (SBERT). SBERT fine-tunes BERT in a siamese / triplet network architecture. We evaluated the quality on various common benchmarks, where it could achieve a significant improvement over state-of-the-art sentence embeddings methods. Replacing BERT with RoBERTa did not yield a significant improvement in our experiments.

SBERT is computationally efficient. On a GPU, it is about 9% faster than InferSent and about 55% faster than Universal Sentence Encoder. SBERT can be used for tasks which are computationally not feasible to be modeled with BERT. For example, clustering of 10,000 sentences with hierarchical clustering requires with BERT about 65 hours, as around 50 Million sentence combinations must be computed. With SBERT, we were able to reduce the effort to about 5 seconds.

[4] https://github.com/facebookresearch/InferSent

[5] https://tfhub.dev/google/universal-sentence-encoder-large/3

Acknowledgments

This work has been supported by the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1 and grant GU 798/17-1). It has been co-funded by the German Federal Ministry of Education and Research (BMBF) under the promotional references 03VP02540 (ArgumenText).

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91, Dublin, Ireland. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 497–511.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic Textual Similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 385–393, Stroudsburg, PA, USA. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Iigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. *arXiv preprint arXiv:1803.11175*.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An Evaluation Toolkit for Universal Sentence Representations. *arXiv preprint arXiv:1803.05449*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liat Ein Dor, Yosi Mass, Alon Halfon, Elad Venezian, Ilya Shnayderman, Ranit Aharonov, and Noam Slonim. 2018. Learning Thematic Similarity Metric from Article Sections Using Triplet Networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54, Melbourne, Australia. Association for Computational Linguistics.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning Distributed Representations of Sentences from Unlabelled Data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, San Diego, California. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Real-time Inference in Multi-sentence Tasks with Deep Pretrained Transformers. *arXiv preprint arXiv:1905.01969*, abs/1905.01969.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc.

Xin Li and Dan Roth. 2002. Learning Question Classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On Measuring Social Biases in Sentence Encoders. *arXiv preprint arXiv:1903.10561*.

Amita Misra, Brian Ecker, and Marilyn A. Walker. 2016. Measuring the Similarity of Sentential Arguments in Dialogue. In *Proceedings of the SIGDIAL 2016 Conference, The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 13-15 September 2016, Los Angeles, CA, USA*, pages 276–287.

Bo Pang and Lillian Lee. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 271–278, Barcelona, Spain.

Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Yifan Qiao, Chenyan Xiong, Zheng-Hao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *arXiv preprint arXiv:1904.07531*.

Nils Reimers, Philip Beyer, and Iryna Gurevych. 2016. Task-Oriented Intrinsic Evaluation of Semantic Textual Similarity. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 87–96.

Nils Reimers and Iryna Gurevych. 2018. Why Comparing Single Performance Scores Does Not Allow to Draw Conclusions About Machine Learning Approaches. *arXiv preprint arXiv:1803.09578*, abs/1803.09578.

Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2019. Classification and Clustering of Arguments with Contextualized Word Embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 567–578, Florence, Italy. Association for Computational Linguistics.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering. *arXiv preprint arXiv:1503.03832*, abs/1503.03832.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating Expressions of Opinions and Emotions in Language. *Language Resources and Evaluation*, 39(2):165–210.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-Yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning Semantic Textual Similarity from Conversations. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 164–174, Melbourne, Australia. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237*, abs/1906.08237.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. BERTScore: Evaluating Text Generation with BERT. *arXiv preprint arXiv:1904.09675*.

Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference

Timo Schick[1,2] **Hinrich Schütze**[1]

[1] Center for Information and Language Processing, LMU Munich, Germany
[2] Sulzer GmbH, Munich, Germany

schickt@cis.lmu.de

Abstract

Some NLP tasks can be solved in a fully unsupervised fashion by providing a pretrained language model with "task descriptions" in natural language (e.g., Radford et al., 2019). While this approach underperforms its supervised counterpart, we show in this work that the two ideas can be combined: We introduce Pattern-Exploiting Training (PET), a semi-supervised training procedure that reformulates input examples as cloze-style phrases to help language models understand a given task. These phrases are then used to assign soft labels to a large set of unlabeled examples. Finally, standard supervised training is performed on the resulting training set. For several tasks and languages, PET outperforms supervised training and strong semi-supervised approaches in low-resource settings by a large margin.[1]

1 Introduction

Learning from examples is the predominant approach for many NLP tasks: A model is trained on a set of labeled examples from which it then generalizes to unseen data. Due to the vast number of languages, domains and tasks and the cost of annotating data, it is common in real-world uses of NLP to have only a small number of labeled examples, making *few-shot learning* a highly important research area. Unfortunately, applying standard supervised learning to small training sets often performs poorly; many problems are difficult to grasp from just looking at a few examples. For instance, assume we are given the following pieces of text:

- T_1: This was the best pizza I've ever had.
- T_2: You can get better sushi for half the price.
- T_3: Pizza was average. Not worth the price.

[1]Our implementation is publicly available at https://github.com/timoschick/pet.

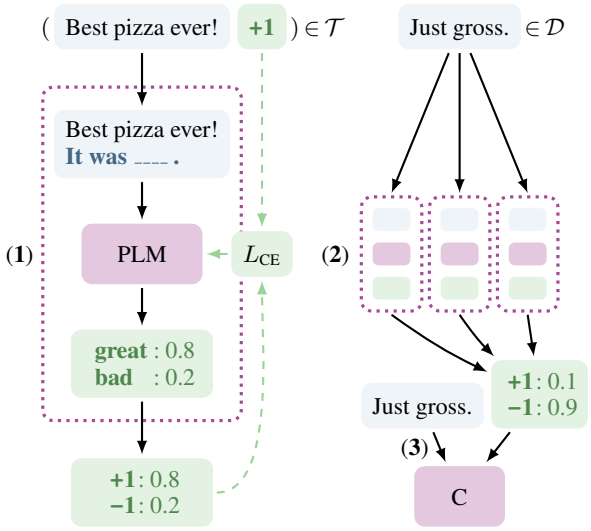


Figure 1: PET for sentiment classification. **(1)** A number of patterns encoding some form of task description are created to convert training examples to cloze questions; for each pattern, a pretrained language model is finetuned. **(2)** The ensemble of trained models annotates unlabeled data. **(3)** A classifier is trained on the resulting soft-labeled dataset.

Furthermore, imagine we are told that the labels of T_1 and T_2 are l and l', respectively, and we are asked to infer the correct label for T_3. Based only on these examples, this is impossible because plausible justifications can be found for both l and l'. However, if we know that the underlying task is to identify whether the text says anything about prices, we can easily assign l' to T_3. This illustrates that solving a task from only a few examples becomes much easier when we also have a *task description*, i.e., a textual explanation that helps us *understand* what the task is about.

With the rise of pretrained language models (PLMs) such as GPT (Radford et al., 2018), BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), the idea of providing task descriptions has become feasible for neural architectures: We can

simply append such descriptions in natural language to an input and let the PLM predict continuations that solve the task (Radford et al., 2019; Puri and Catanzaro, 2019). So far, this idea has mostly been considered in zero-shot scenarios where no training data is available at all.

In this work, we show that providing task descriptions can successfully be combined with standard supervised learning in few-shot settings: We introduce **P**attern-**E**xploiting **T**raining (PET), a semi-supervised training procedure that uses natural language patterns to reformulate input examples into cloze-style phrases. As illustrated in Figure 1, PET works in three steps: First, for each pattern a separate PLM is finetuned on a small training set $\mathcal{T}$. The ensemble of all models is then used to annotate a large unlabeled dataset $\mathcal{D}$ with soft labels. Finally, a standard classifier is trained on the soft-labeled dataset. We also devise iPET, an iterative variant of PET in which this process is repeated with increasing training set sizes.

On a diverse set of tasks in multiple languages, we show that given a small to medium number of labeled examples, PET and iPET substantially outperform unsupervised approaches, supervised training and strong semi-supervised baselines.

2 Related Work

Radford et al. (2019) provide hints in the form of natural language patterns for zero-shot learning of challenging tasks such as reading comprehension and question answering (QA). This idea has been applied to unsupervised text classification (Puri and Catanzaro, 2019), commonsense knowledge mining (Davison et al., 2019) and argumentative relation classification (Opitz, 2019). Srivastava et al. (2018) use task descriptions for zero-shot classification but require a semantic parser. For relation extraction, Bouraoui et al. (2020) automatically identify patterns that express given relations. McCann et al. (2018) rephrase several tasks as QA problems. Raffel et al. (2020) frame various problems as language modeling tasks, but their patterns only loosely resemble natural language and are unsuitable for few-shot learning.[2]

Another recent line of work uses cloze-style phrases to probe the knowledge that PLMs acquire during pretraining; this includes probing for factual and commonsense knowledge (Trinh and Le, 2018; Petroni et al., 2019; Wang et al., 2019; Sakaguchi et al., 2020), linguistic capabilities (Ettinger, 2020; Kassner and Schütze, 2020), understanding of rare words (Schick and Schütze, 2020), and ability to perform symbolic reasoning (Talmor et al., 2019). Jiang et al. (2020) consider the problem of finding the best pattern to express a given task.

Other approaches for few-shot learning in NLP include exploiting examples from related tasks (Yu et al., 2018; Gu et al., 2018; Dou et al., 2019; Qian and Yu, 2019; Yin et al., 2019) and using data augmentation (Xie et al., 2020; Chen et al., 2020); the latter commonly relies on back-translation (Sennrich et al., 2016), requiring large amounts of parallel data. Approaches using textual class descriptors typically assume that abundant examples are available for a subset of classes (e.g., Romera-Paredes and Torr, 2015; Veeranna et al., 2016; Ye et al., 2020). In contrast, our approach requires no additional labeled data and provides an intuitive interface to leverage task-specific human knowledge.

The idea behind iPET – training multiple generations of models on data labeled by previous generations – bears resemblance to self-training and bootstrapping approaches for word sense disambiguation (Yarowsky, 1995), relation extraction (Brin, 1999; Agichtein and Gravano, 2000; Batista et al., 2015), parsing (McClosky et al., 2006; Reichart and Rappoport, 2007; Huang and Harper, 2009), machine translation (Hoang et al., 2018), and sequence generation (He et al., 2020).

3 Pattern-Exploiting Training

Let M be a masked language model with vocabulary V and mask token $\_\_\_\_ \in V$, and let $\mathcal{L}$ be a set of labels for our target classification task A. We write an input for task A as a sequence of phrases $\mathbf{x} = (s_1, \ldots, s_k)$ with $s_i \in V^*$; for example, $k = 2$ if A is textual inference (two input sentences). We define a *pattern* to be a function P that takes $\mathbf{x}$ as input and outputs a phrase or sentence $P(\mathbf{x}) \in V^*$ that contains exactly one mask token, i.e., its output can be viewed as a cloze question. Furthermore, we define a *verbalizer* as an injective function $v : \mathcal{L} \rightarrow V$ that maps each label to a word from M's vocabulary. We refer to (P, v) as a *pattern-verbalizer pair* (PVP).

Using a PVP (P, v) enables us to solve task A as follows: Given an input $\mathbf{x}$, we apply P to obtain an input representation $P(\mathbf{x})$, which is then processed

[2] For example, they convert inputs (a, b) for recognizing textual entailment (RTE) to "rte sentence1: a sentence2: b", and the PLM is asked to predict strings like "not_entailment".

by M to determine the label $y \in \mathcal{L}$ for which $v(y)$ is the most likely substitute for the mask. For example, consider the task of identifying whether two sentences a and b contradict each other (label y_0) or agree with each other (y_1). For this task, we may choose the pattern $P(a, b) =$ a? ____, b. combined with a verbalizer v that maps y_0 to "Yes" and y_1 to "No". Given an example input pair

$$\mathbf{x} = (\text{Mia likes pie, Mia hates pie}),$$

the task now changes from having to assign a label without inherent meaning to answering whether the most likely choice for the masked position in

$$P(\mathbf{x}) = \text{Mia likes pie? ____, Mia hates pie.}$$

is "Yes" or "No".

3.1 PVP Training and Inference

Let $\mathbf{p} = (P, v)$ be a PVP. We assume access to a small training set $\mathcal{T}$ and a (typically much larger) set of unlabeled examples $\mathcal{D}$. For each sequence $\mathbf{z} \in V^*$ that contains exactly one mask token and $w \in V$, we denote with $M(w \mid \mathbf{z})$ the unnormalized score that the language model assigns to w at the masked position. Given some input $\mathbf{x}$, we define the score for label $l \in \mathcal{L}$ as

$$s_{\mathbf{p}}(l \mid \mathbf{x}) = M(v(l) \mid P(\mathbf{x}))$$

and obtain a probability distribution over labels using softmax:

$$q_{\mathbf{p}}(l \mid \mathbf{x}) = \frac{e^{s_{\mathbf{p}}(l|\mathbf{x})}}{\sum_{l' \in \mathcal{L}} e^{s_{\mathbf{p}}(l'|\mathbf{x})}}$$

We use the cross-entropy between $q_{\mathbf{p}}(l \mid \mathbf{x})$ and the true (one-hot) distribution of training example $(\mathbf{x}, l)$ – summed over all $(\mathbf{x}, l) \in \mathcal{T}$ – as loss for finetuning M for $\mathbf{p}$.

3.2 Auxiliary Language Modeling

In our application scenario, only a few training examples are available and catastrophic forgetting can occur. As a PLM finetuned for some PVP is still a language model at its core, we address this by using language modeling as auxiliary task. With L_{CE} denoting cross-entropy loss and L_{MLM} language modeling loss, we compute the final loss as

$$L = (1 - \alpha) \cdot L_{\text{CE}} + \alpha \cdot L_{\text{MLM}}$$

This idea was recently applied by Chronopoulou et al. (2019) in a data-rich scenario. As L_{MLM} is typically much larger than L_{CE}, in preliminary experiments, we found a small value of $\alpha = 10^{-4}$ to consistently give good results, so we use it in all our experiments. To obtain sentences for language modeling, we use the unlabeled set $\mathcal{D}$. However, we do not train directly on each $\mathbf{x} \in \mathcal{D}$, but rather on $P(\mathbf{x})$, where we never ask the language model to predict anything for the masked slot.

3.3 Combining PVPs

A key challenge for our approach is that in the absence of a large development set, it is hard to identify which PVPs perform well. To address this, we use a strategy similar to knowledge distillation (Hinton et al., 2015). First, we define a set $\mathcal{P}$ of PVPs that intuitively make sense for a given task A. We then use these PVPs as follows:

(1) We finetune a separate language model $M_{\mathbf{p}}$ for each $\mathbf{p} \in \mathcal{P}$ as described in Section 3.1. As $\mathcal{T}$ is small, this finetuning is cheap even for a large number of PVPs.

(2) We use the ensemble $\mathcal{M} = \{M_{\mathbf{p}} \mid \mathbf{p} \in \mathcal{P}\}$ of finetuned models to annotate examples from $\mathcal{D}$. We first combine the unnormalized class scores for each example $\mathbf{x} \in \mathcal{D}$ as

$$s_{\mathcal{M}}(l \mid \mathbf{x}) = \frac{1}{Z} \sum_{\mathbf{p} \in \mathcal{P}} w(\mathbf{p}) \cdot s_{\mathbf{p}}(l \mid \mathbf{x})$$

where $Z = \sum_{\mathbf{p} \in \mathcal{P}} w(\mathbf{p})$ and the $w(\mathbf{p})$ are weighting terms for the PVPs. We experiment with two different realizations of this weighing term: either we simply set $w(\mathbf{p}) = 1$ for all $\mathbf{p}$ or we set $w(\mathbf{p})$ to be the accuracy obtained using $\mathbf{p}$ on the training set *before* training. We refer to these two variants as *uniform* and *weighted*. Jiang et al. (2020) use a similar idea in a zero-shot setting.

We transform the above scores into a probability distribution q using softmax. Following Hinton et al. (2015), we use a temperature of $T = 2$ to obtain a suitably soft distribution. All pairs $(\mathbf{x}, q)$ are collected in a (soft-labeled) training set $\mathcal{T}_C$.

(3) We finetune a PLM C with a standard sequence classification head on $\mathcal{T}_C$.

The finetuned model C then serves as our classifier for A. All steps described above are depicted in Figure 2; an example is shown in Figure 1.

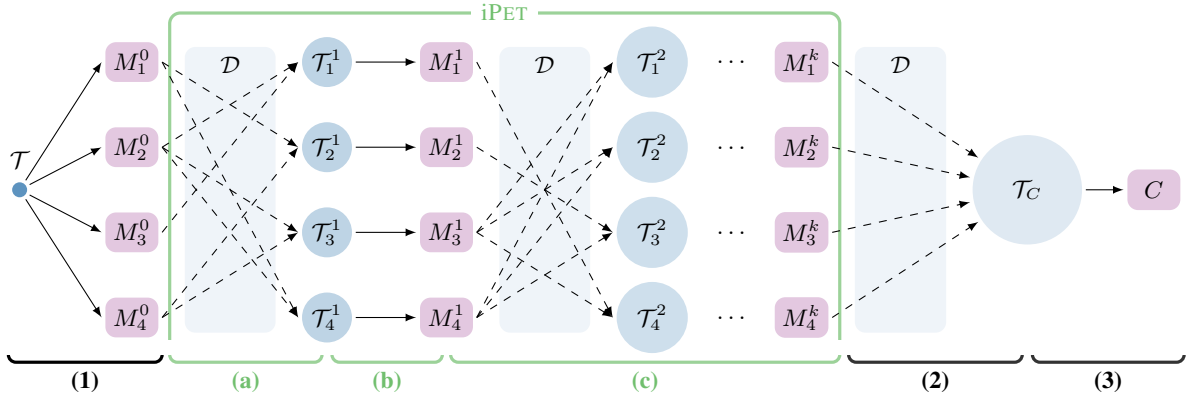


Figure 2: Schematic representation of PET (1-3) and iPET (a-c). **(1)** The initial training set is used to finetune an ensemble of PLMs. **(a)** For each model, a random subset of other models generates a new training set by labeling examples from $\mathcal{D}$. **(b)** A new set of PET models is trained using the larger, model-specific datasets. **(c)** The previous two steps are repeated k times, each time increasing the size of the generated training sets by a factor of d. **(2)** The final set of models is used to create a soft-labeled dataset $\mathcal{T}_C$. **(3)** A classifier C is trained on this dataset.

3.4 Iterative PET (iPET)

Distilling the knowledge of all individual models into a single classifier C means they cannot learn from each other. As some patterns perform (possibly much) worse than others, the training set $\mathcal{T}_C$ for our final model may therefore contain many mislabeled examples.

To compensate for this shortcoming, we devise iPET, an iterative variant of PET. The core idea of iPET is to train several *generations* of models on datasets of increasing size. To this end, we first enlarge the original dataset $\mathcal{T}$ by labeling selected examples from $\mathcal{D}$ using a random subset of trained PET models (Figure 2a). We then train a new generation of PET models on the enlarged dataset (b); this process is repeated several times (c).

More formally, let $\mathcal{M}^0 = \{M_1^0, \ldots, M_n^0\}$ be the initial set of PET models finetuned on $\mathcal{T}$, where each M_i^0 is trained for some PVP $\mathbf{p}_i$. We train k generations of models $\mathcal{M}^1, \ldots, \mathcal{M}^k$ where $\mathcal{M}^j = \{M_1^j, \ldots, M_n^j\}$ and each M_i^j is trained for $\mathbf{p}_i$ on its own training set $\mathcal{T}_i^j$. In each iteration, we multiply the training set size by a fixed constant $d \in \mathbb{N}$ while maintaining the label ratio of the original dataset. That is, with $c_0(l)$ denoting the number of examples with label l in $\mathcal{T}$, each $\mathcal{T}_i^j$ contains $c_j(l) = d \cdot c_{j-1}(l)$ examples with label l. This is achieved by generating each $\mathcal{T}_i^j$ as follows:

1. We obtain $\mathcal{N} \subset \mathcal{M}^{j-1} \setminus \{M_i^{j-1}\}$ by randomly choosing $\lambda \cdot (n-1)$ models from the previous generation with $\lambda \in (0, 1]$ being a hyperparameter.

2. Using this subset, we create a labeled dataset

$$\mathcal{T}_{\mathcal{N}} = \{(\mathbf{x}, \arg\max_{l \in \mathcal{L}} s_{\mathcal{N}}(l \mid \mathbf{x})) \mid \mathbf{x} \in \mathcal{D}\}.$$

   For each $l \in \mathcal{L}$, we obtain $\mathcal{T}_{\mathcal{N}}(l) \subset \mathcal{T}_{\mathcal{N}}$ by randomly choosing $c_j(l) - c_0(l)$ examples with label l from $\mathcal{T}_{\mathcal{N}}$. To avoid training future generations on mislabeled data, we prefer examples for which the ensemble of models is confident in its prediction. The underlying intuition is that even without calibration, examples for which labels are predicted with high confidence are typically more likely to be classified correctly (Guo et al., 2017). Therefore, when drawing from $\mathcal{T}_{\mathcal{N}}$, we set the probability of each $(\mathbf{x}, y)$ proportional to $s_{\mathcal{N}}(l \mid \mathbf{x})$.

3. We define $\mathcal{T}_i^j = \mathcal{T} \cup \bigcup_{l \in \mathcal{L}} \mathcal{T}_{\mathcal{N}}(l)$. As can easily be verified, this dataset contains $c_j(l)$ examples for each $l \in \mathcal{L}$.

After training k generations of PET models, we use $\mathcal{M}^k$ to create $\mathcal{T}_C$ and train C as in basic PET.

With minor adjustments, iPET can even be used in a zero-shot setting. To this end, we define $\mathcal{M}^0$ to be the set of *untrained* models and $c_1(l) = 10/|\mathcal{L}|$ for all $l \in \mathcal{L}$ so that $\mathcal{M}^1$ is trained on 10 examples evenly distributed across all labels. As $\mathcal{T}_{\mathcal{N}}$ may not contain enough examples for some label l, we create all $\mathcal{T}_{\mathcal{N}}(l)$ by sampling from the 100 examples $\mathbf{x} \in \mathcal{D}$ for which $s_{\mathcal{N}}(l \mid x)$ is the highest, even if $l \neq \arg\max_{l \in \mathcal{L}} s_{\mathcal{N}}(l \mid x)$. For each subsequent generation, we proceed exactly as in basic iPET.

4 Experiments

We evaluate PET on four English datasets: Yelp Reviews, AG's News, Yahoo Questions (Zhang et al., 2015) and MNLI (Williams et al., 2018). Additionally, we use x-stance (Vamvas and Sennrich, 2020) to investigate how well PET works for other languages. For all experiments on English, we use RoBERTa large (Liu et al., 2019) as language model; for x-stance, we use XLM-R (Conneau et al., 2020). We investigate the performance of PET and all baselines for different training set sizes; each model is trained three times using different seeds and average results are reported.

As we consider a few-shot setting, we assume no access to a large development set on which hyperparameters could be optimized. Our choice of hyperparameters is thus based on choices made in previous work and practical considerations. We use a learning rate of $1 \cdot 10^{-5}$, a batch size of 16 and a maximum sequence length of 256. Unless otherwise specified, we always use the weighted variant of PET with auxiliary language modeling. For iPET, we set $\lambda = 0.25$ and $d = 5$; that is, we select 25% of all models to label examples for the next generation and quintuple the number of training examples in each iteration. We train new generations until each model was trained on at least 1000 examples, i.e., we set $k = \lceil \log_d(1000/|\mathcal{T}|) \rceil$. As we always repeat training three times, the ensemble $\mathcal{M}$ (or $\mathcal{M}^0$) for n PVPs contains $3n$ models. Further hyperparameters and detailed explanations for all our choices are given in Appendix B.

4.1 Patterns

We now describe the patterns and verbalizers used for all tasks. We use two vertical bars ($\|$) to mark boundaries between text segments.[3]

Yelp For the Yelp Reviews Full Star dataset (Zhang et al., 2015), the task is to estimate the rating that a customer gave to a restaurant on a 1- to 5-star scale based on their review's text. We define the following patterns for an input text a:

$P_1(a) =$ It was ____. a $\quad P_2(a) =$ Just ____! $\| \; a$

$P_3(a) =$ a. All in all, it was ____.

$P_4(a) =$ $a \; \|$ In summary, the restaurant is ____.

We define a single verbalizer v for all patterns as

$v(1) =$ terrible $\quad v(2) =$ bad $\quad v(3) =$ okay

$v(4) =$ good $\quad v(5) =$ great

AG's News AG's News is a news classification dataset, where given a headline a and text body b, news have to be classified as belonging to one of the categories *World* (1), *Sports* (2), *Business* (3) or *Science/Tech* (4). For $\mathbf{x} = (a, b)$, we define the following patterns:

$P_1(\mathbf{x}) =$ ____: a b $\quad P_2(\mathbf{x}) =$ a (____) b

$P_3(\mathbf{x}) =$ ____ – a b $\quad P_4(\mathbf{x}) =$ a b (____)

$P_5(\mathbf{x}) =$ ____ News: a b

$P_6(\mathbf{x}) =$ [Category: ____] a b

We use a verbalizer that maps 1–4 to "World", "Sports", "Business" and "Tech", respectively.

Yahoo Yahoo Questions (Zhang et al., 2015) is a text classification dataset. Given a question a and an answer b, one of ten possible categories has to be assigned. We use the same patterns as for AG's News, but we replace the word "News" in P_5 with the word "Question". We define a verbalizer that maps categories 1–10 to "Society", "Science", "Health", "Education", "Computer", "Sports", "Business", "Entertainment", "Relationship" and "Politics".

MNLI The MNLI dataset (Williams et al., 2018) consists of text pairs $\mathbf{x} = (a, b)$. The task is to find out whether a implies b (0), a and b contradict each other (1) or neither (2). We define

$P_1(\mathbf{x}) =$ "a"? $\|$ ____, "b" $\quad P_2(\mathbf{x}) =$ a? $\|$ ____, b

and consider two different verbalizers v_1 and v_2:

$v_1(0) =$ Wrong $\quad v_1(1) =$ Right $\quad v_1(2) =$ Maybe

$v_2(0) =$ No $\quad v_2(1) =$ Yes $\quad v_2(2) =$ Maybe

Combining the two patterns with the two verbalizers results in a total of 4 PVPs.

X-Stance The x-stance dataset (Vamvas and Sennrich, 2020) is a multilingual stance detection dataset with German, French and Italian examples. Each example $\mathbf{x} = (a, b)$ consists of a question a concerning some political issue and a comment b; the task is to identify whether the writer of b

[3] The way different segments are handled depends on the model being used; they may e.g. be assigned different embeddings (Devlin et al., 2019) or separated by special tokens (Liu et al., 2019; Yang et al., 2019). For example, "$a \; \| \; b$" is given to BERT as the input "[CLS] a [SEP] b [SEP]".

Line	Examples	Method	Yelp	AG's	Yahoo	MNLI (m/mm)
1	$\|\mathcal{T}\| = 0$	unsupervised (avg)	33.8 ±9.6	69.5 ±7.2	44.0 ±9.1	39.1 ±4.3 / 39.8 ±5.1
2		unsupervised (max)	40.8 ±0.0	79.4 ±0.0	56.4 ±0.0	43.8 ±0.0 / 45.0 ±0.0
3		iPET	**56.7** ±0.2	**87.5** ±0.1	**70.7** ±0.1	**53.6** ±0.1 / **54.2** ±0.1
4	$\|\mathcal{T}\| = 10$	supervised	21.1 ±1.6	25.0 ±0.1	10.1 ±0.1	34.2 ±2.1 / 34.1 ±2.0
5		PET	52.9 ±0.1	87.5 ±0.0	63.8 ±0.2	41.8 ±0.1 / 41.5 ±0.2
6		iPET	**57.6** ±0.0	**89.3** ±0.1	**70.7** ±0.1	**43.2** ±0.0 / **45.7** ±0.1
7	$\|\mathcal{T}\| = 50$	supervised	44.8 ±2.7	82.1 ±2.5	52.5 ±3.1	45.6 ±1.8 / 47.6 ±2.4
8		PET	60.0 ±0.1	86.3 ±0.0	66.2 ±0.1	63.9 ±0.0 / 64.2 ±0.0
9		iPET	**60.7** ±0.1	**88.4** ±0.1	**69.7** ±0.0	**67.4** ±0.3 / **68.3** ±0.3
10	$\|\mathcal{T}\| = 100$	supervised	53.0 ±3.1	86.0 ±0.7	62.9 ±0.9	47.9 ±2.8 / 51.2 ±2.6
11		PET	61.9 ±0.0	88.3 ±0.1	69.2 ±0.0	74.7 ±0.3 / 75.9 ±0.4
12		iPET	**62.9** ±0.0	**89.6** ±0.1	**71.2** ±0.1	**78.4** ±0.7 / **78.6** ±0.5
13	$\|\mathcal{T}\| = 1000$	supervised	63.0 ±0.5	**86.9** ±0.4	70.5 ±0.3	73.1 ±0.2 / 74.8 ±0.3
14		PET	**64.8** ±0.1	**86.9** ±0.2	**72.7** ±0.0	**85.3** ±0.2 / **85.5** ±0.4

Table 1: Average accuracy and standard deviation for RoBERTa (large) on Yelp, AG's News, Yahoo and MNLI (m:matched/mm:mismatched) for five training set sizes $|\mathcal{T}|$.

supports the subject of the question (0) or not (1). We use two simple patterns

$$P_1(\mathbf{x}) = \text{“}a\text{”} \parallel \text{\_\_\_\_\_. “}b\text{”} \qquad P_2(\mathbf{x}) = a \parallel \text{\_\_\_\_\_. } b$$

and define an English verbalizer v_{En} mapping 0 to “Yes” and 1 to “No” as well as a French (German) verbalizer v_{Fr} (v_{De}), replacing “Yes” and “No” with “Oui” and “Non” (“Ja” and “Nein”). We do not define an Italian verbalizer because x-stance does not contain any Italian training examples.

4.2 Results

English Datasets Table 1 shows results for English text classification and language understanding tasks; we report mean accuracy and standard deviation for three training runs. Lines 1–2 (L1–L2) show unsupervised performance, i.e., individual PVPs without *any* training (similar to Radford et al., 2018; Puri and Catanzaro, 2019); we give both average results across all PVPs (avg) and results for the PVP that works best on the test set (max). The large difference between both rows highlights the importance of coping with the fact that without looking at the test set, we have no means of evaluating which PVPs perform well. Zero-shot iPET clearly outperforms the unsupervised baselines for all datasets (L3 vs L1); on AG's News, it even performs better than standard supervised training with 1000 examples (L3 vs L13). With just 10 training examples, standard supervised learning does not perform above chance (L4). In contrast, PET (L5) performs much better than the fully unsupervised baselines (L1–L2); training multiple generations using iPET (L6) gives consistent improvements. As we increase the training set size, the performance gains of PET and iPET become smaller, but for both 50 and 100 examples, PET continues to considerably outperform standard supervised training (L8 vs L7, L11 vs L10) with iPET (L9, L12) still giving consistent improvements. For $|\mathcal{T}| = 1000$, PET has no advantage on AG's but still improves accuracy for all other tasks (L14 vs L13).[4]

Ex.	Method	Yelp	AG's	Yahoo	MNLI
$\|\mathcal{T}\| = 10$	UDA	27.3	72.6	36.7	34.7
	MixText	20.4	81.1	20.6	32.9
	PET	48.8	84.1	59.0	39.5
	iPET	**52.9**	**87.5**	**67.0**	**42.1**
$\|\mathcal{T}\| = 50$	UDA	46.6	83.0	60.2	40.8
	MixText	31.3	84.8	61.5	34.8
	PET	55.3	86.4	63.3	55.1
	iPET	**56.7**	**87.3**	**66.4**	**56.3**

Table 2: Comparison of PET with two state-of-the-art semi-supervised methods using RoBERTa (base)

Comparison with SotA We compare PET to UDA (Xie et al., 2020) and MixText (Chen et al., 2020), two state-of-the-art methods for semi-supervised learning in NLP that rely on data augmentation. Whereas PET requires that a task can be expressed using patterns and that such patterns be found, UDA and MixText both use backtranslation (Sennrich et al., 2016) and thus require thousands of labeled examples for training a machine translation model. We use RoBERTa (base) for our comparison as MixText is specifically tailored towards

[4] One of the three supervised MNLI runs for $|\mathcal{T}| = 1000$ underfitted the training data and performed extremely poorly. This run is excluded in the reported score (73.1/74.8).

Examples	Method	De	Fr	It
$\lvert\mathcal{T}\rvert = 1000$	supervised	43.3	49.5	41.0
	PET	**66.4**	**68.7**	**64.7**
$\lvert\mathcal{T}\rvert = 2000$	supervised	57.4	62.1	52.8
	PET	**69.5**	**71.7**	**67.3**
$\lvert\mathcal{T}\rvert = 4000$	supervised	63.2	66.7	58.7
	PET	**71.7**	**74.0**	**69.5**
$\mathcal{T}_{\text{De}}$, $\mathcal{T}_{\text{Fr}}$	supervised	76.6	76.0	71.0
	PET	**77.9**	**79.0**	**73.6**
$\mathcal{T}_{\text{De}} + \mathcal{T}_{\text{Fr}}$	sup. (*)	76.8	76.7	70.2
	supervised	77.6	79.1	75.9
	PET	**78.8**	**80.6**	**77.2**

Table 3: Results on x-stance intra-target for XLM-R (base) trained on subsets of $\mathcal{T}_{\text{De}}$ and $\mathcal{T}_{\text{Fr}}$ and for joint training on all data ($\mathcal{T}_{\text{De}} + \mathcal{T}_{\text{Fr}}$). (*): Best results for mBERT reported in Vamvas and Sennrich (2020).

a 12-layer Transformer (Vaswani et al., 2017). Both Xie et al. (2020) and Chen et al. (2020) use large development sets to optimize the number of training steps. We instead try several values for both approaches directly on the test set and only report the *best* results obtained. Despite this, Table 2 shows that PET and iPET substantially outperform both methods across all tasks, clearly demonstrating the benefit of incorporating human knowledge in the form of PVPs.

X-Stance We evaluate PET on x-stance to investigate (i) whether it works for languages other than English and (ii) whether it also brings improvements when training sets have medium size. In contrast to Vamvas and Sennrich (2020), we do not perform any hyperparameter optimization on dev and use a shorter maximum sequence length (256 vs 512) to speed up training and evaluation.

To investigate whether PET brings benefits even when numerous examples are available, we consider training set sizes of 1000, 2000, and 4000; for each of these configurations, we separately finetune French and German models to allow for a more straightforward downsampling of the training data. Additionally, we train models on the entire French ($\lvert\mathcal{T}_{\text{Fr}}\rvert = 11\,790$) and German ($\lvert\mathcal{T}_{\text{De}}\rvert = 33\,850$) training sets. In this case we do not have any additional unlabeled data, so we simply set $\mathcal{D} = \mathcal{T}$. For the French models, we use v_{En} and v_{Fr} as verbalizers and for German v_{En} and v_{De} (Section 4.1). Finally, we also investigate the performance of a model trained jointly on French and German data ($\lvert\mathcal{T}_{\text{Fr}} + \mathcal{T}_{\text{De}}\rvert = 45\,640$) using v_{En}, v_{Fr} and v_{De}.

Results are shown in Table 3; following Vamvas and Sennrich (2020), we report the macro-average of the F1 scores for labels 0 and 1, averaged over three runs. For Italian (column "It"), we report the average zero-shot cross-lingual performance of German and French models as there are no Italian training examples. Our results show that PET brings huge improvements across all languages even when training on much more than a thousand examples; it also considerably improves zero-shot cross-lingual performance.

Method	Yelp	AG's	Yahoo	MNLI
min	39.6	82.1	50.2	36.4
max	52.4	85.0	63.6	40.2
PET (no distillation)	51.7	87.0	62.8	40.6
PET uniform	52.7	87.3	**63.8**	**42.0**
PET weighted	**52.9**	**87.5**	**63.8**	41.8

Table 4: Minimum (min) and maximum (max) accuracy of models based on individual PVPs as well as PET with and without knowledge distillation ($\lvert\mathcal{T}\rvert = 10$).

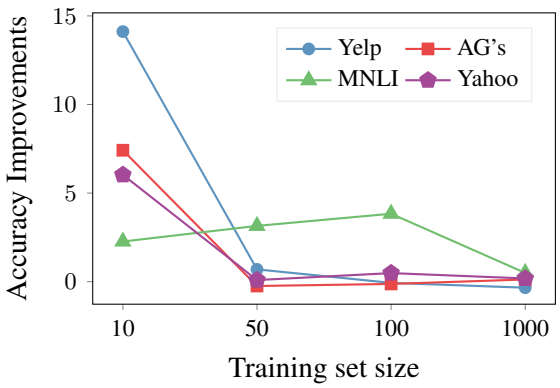


Figure 3: Accuracy improvements for PET due to adding L_{MLM} during training

5 Analysis

Combining PVPs We first investigate whether PET is able to cope with situations were some PVPs perform much worse than others. For $\lvert\mathcal{T}\rvert = 10$, Table 4 compares the performance of PET to that of the best and worst performing patterns after finetuning; we also include results obtained using the ensemble of PET models corresponding to individual PVPs without knowledge distillation. Even after finetuning, the gap between the best and worst pattern is large, especially for Yelp. However, PET is not only able to compensate for this, but even improves accuracies over using only the best-performing pattern across all tasks. Distillation brings consistent improvements over the ensemble; additionally, it significantly reduces the size of the

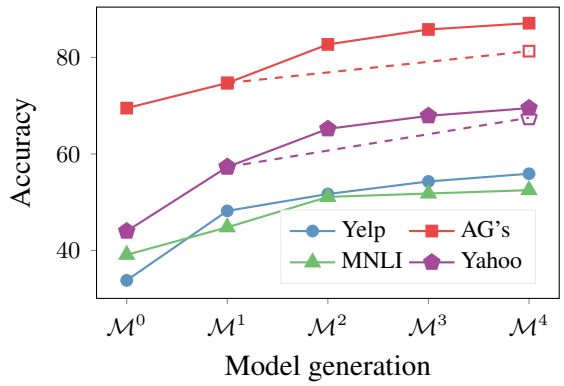


Figure 4: Average accuracy for each generation of models with iPET in a zero-shot setting. Accuracy on AG's News and Yahoo when skipping generation 2 and 3 is indicated through dashed lines.

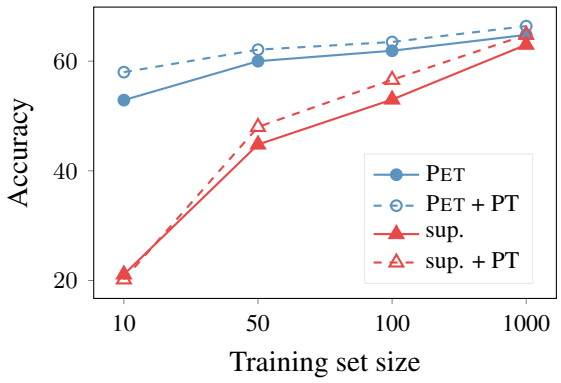


Figure 5: Accuracy of supervised learning (sup.) and PET both with and without pretraining (PT) on Yelp

final classifier. We find no clear difference between the uniform and weighted variants of PET.

Auxiliary Language Modeling We analyze the influence of the auxiliary language modeling task on PET's performance. Figure 3 shows performance improvements from adding the language modeling task for four training set sizes. We see that the auxiliary task is extremely valuable when training on just 10 examples. With more data, it becomes less important, sometimes even leading to worse performance. Only for MNLI, we find language modeling to consistently help.

Iterative PET To check whether iPET is able to improve models over multiple generations, Figure 4 shows the average performance of all generations of models in a zero-shot setting. Each additional iteration does indeed further improve the ensemble's performance. We did not investigate whether continuing this process for even more iterations gives further improvements.

Another natural question is whether similar results can be obtained with fewer iterations by increasing the training set size more aggressively. To answer this question, we skip generations 2 and 3 for AG's News and Yahoo and for both tasks directly let ensemble $\mathcal{M}^1$ annotate $10 \cdot 5^4$ examples for $\mathcal{M}^4$. As indicated in Figure 4 through dashed lines, this clearly leads to worse performance, highlighting the importance of only gradually increasing the training set size. We surmise that this is the case because annotating too many examples too early leads to a large percentage of mislabeled training examples.

In-Domain Pretraining Unlike our supervised baseline, PET makes use of the additional unlabeled dataset $\mathcal{D}$. Thus, at least some of PET's performance gains over the supervised baseline may arise from this additional in-domain data.

To test this hypothesis, we simply further pretrain RoBERTa on in-domain data, a common technique for improving text classification accuracy (e.g., Howard and Ruder, 2018; Sun et al., 2019). As language model pretraining is expensive in terms of GPU usage, we do so only for the Yelp dataset. Figure 5 shows results of supervised learning and PET both with and without this in-domain pretraining. While pretraining does indeed improve accuracy for supervised training, the supervised model still clearly performs worse than PET, showing that the success of our method is not simply due to the usage of additional unlabeled data. Interestingly, in-domain pretraining is also helpful for PET, indicating that PET leverages unlabeled data in a way that is clearly different from standard masked language model pretraining.

6 Conclusion

We have shown that providing task descriptions to pretrained language models can be combined with standard supervised training. Our proposed method, PET, consists of defining pairs of cloze question patterns and verbalizers that help leverage the knowledge contained within pretrained language models for downstream tasks. We finetune models for all pattern-verbalizer pairs and use them to create large annotated datasets on which standard classifiers can be trained. When the initial amount of training data is limited, PET gives large improvements over standard supervised training and strong semi-supervised approaches.

Acknowledgments

This work was funded by the European Research Council (ERC #740516). We would like to thank the anonymous reviewers for their helpful comments.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, page 85–94, New York, NY, USA. Association for Computing Machinery.

David S. Batista, Bruno Martins, and Mário J. Silva. 2015. Semi-supervised bootstrapping of relationship extractors with distributional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 499–504, Lisbon, Portugal. Association for Computational Linguistics.

Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from BERT. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. In *The World Wide Web and Databases*, pages 172–183, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157, Online. Association for Computational Linguistics.

Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. 2019. An embarrassingly simple approach for transfer learning from pretrained language models. 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 2089–2095, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Joe Davison, Joshua Feldman, and Alexander Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178, Hong Kong, China. Association for Computational Linguistics.

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.

Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. 2019. Investigating meta-learning algorithms for low-resource natural language understanding tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1192–1197, Hong Kong, China. Association for Computational Linguistics.

Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631, Brussels, Belgium. Association for Computational Linguistics.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1321–1330. JMLR.org.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *International Conference on Learning Representations*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Computing Research Repository*, arXiv:1503.02531.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers), pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Zhongqiang Huang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 832–841, Singapore. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Nora Kassner and Hinrich Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *Computing Research Repository*, arXiv:1907.11692.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *Computing Research Repository*, arXiv:1806.08730.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA. Association for Computational Linguistics.

Juri Opitz. 2019. Argumentative relation classification as plausibility ranking. In *Preliminary proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers*, pages 193–202, Erlangen, Germany. German Society for Computational Linguistics & Language Technology.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Raul Puri and Bryan Catanzaro. 2019. Zero-shot text classification with generative language models. *Computing Research Repository*, arXiv:1912.10165.

Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2639–2649, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 616–623, Prague, Czech Republic. Association for Computational Linguistics.

Bernardino Romera-Paredes and Philip Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Timo Schick and Hinrich Schütze. 2020. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2018. Zero-shot learning of classifiers from natural language quantification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 306–316, Melbourne, Australia. Association for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? In *Chinese Computational Linguistics*, pages 194–206, Cham. Springer International Publishing.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2019. oLMpics – on what language model pre-training captures. *Computing Research Repository*, arXiv:1912.13283.

Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning. *Computing Research Repository*, arXiv:1806.02847.

Jannis Vamvas and Rico Sennrich. 2020. X-stance: A multilingual multi-target dataset for stance detection. *Computing Research Repository*, arXiv:2003.08385.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Sappadla Prateek Veeranna, Jinseok Nam, Eneldo Loza Mencıa, and Johannes Fürnkranz. 2016. Using semantic similarity for multi-label zero-shot classification of text documents. In *Proceeding of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges, Belgium: Elsevier*, pages 423–428.

Cunxiang Wang, Shuailong Liang, Yue Zhang, Xiaonan Li, and Tian Gao. 2019. Does it make sense? And why? A pilot study for sense making and explanation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4020–4026, Florence, Italy. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2020. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Zhiquan Ye, Yuxia Geng, Jiaoyan Chen, Jingmin Chen, Xiaoxiao Xu, SuHang Zheng, Feng Wang, Jun Zhang, and Huajun Chen. 2020. Zero-shot text classification via reinforced self-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3014–3024, Online. Association for Computational Linguistics.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3914–3923, Hong Kong, China. Association for Computational Linguistics.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1206–1215, New Orleans, Louisiana. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

A Implementation

Our implementation of PET and iPET is based on the Transformers library (Wolf et al., 2020) and PyTorch (Paszke et al., 2017).

B Training Details

Except for the in-domain pretraining experiment described in Section 5, all of our experiments were conducted using a single GPU with 11GB RAM (NVIDIA GeForce GTX 1080 Ti).

B.1 Hyperparameter Choices

Relevant training hyperparameters for both individual PET models and the final classifier C as well as our supervised baseline are listed in Table 5. All hyperparameters were selected based on the following considerations and experiments:

Batch size / maximum length Both batch size and maximum sequence length (or block size) are chosen so that one batch fits into 11GB of GPU memory. As Devlin et al. (2019) and Liu et al. (2019) use larger batch sizes of 16–32, we accumulate gradients for 4 steps to obtain an effective batch size of 16.

Learning rate We found a learning rate of $5\text{e}{-5}$ (as used by Devlin et al. (2019)) to often result in unstable training for regular supervised learning with no accuracy improvements on the training set. We therefore use a lower learning rate of $1\text{e}{-5}$, similar to Liu et al. (2019). Experiments with various learning rates can be found in Appendix D.

Training steps As the number of training epochs recommended by Liu et al. (2019) in a data-rich scenario is in the range 2–10, we perform supervised training for 250 training steps, corresponding to 4 epochs when training on 1000 examples. For individual PET models, we subdivide each batch into one labeled example from $\mathcal{T}$ to compute L_{CE} and three unlabeled examples from $\mathcal{D}$ to compute L_{MLM}. Accordingly, we multiply the number of total training steps by 4 (i.e., 1000), so that the number of times each labeled example is seen remains constant ($16 \cdot 250 = 4 \cdot 1000$). For the final PET classifier, we train for 5000 steps due to the increased training set size (depending on the task, the unlabeled set $\mathcal{D}$ contains at least 20 000 examples). Deviating from the above, we always perform training for 3 epochs on x-stance to match the setup of Vamvas and Sennrich (2020) more closely. The effect of varying the number of training steps is further investigated in Appendix D.

Temperature We choose a temperature of 2 when training the final classifier following Hinton et al. (2015).

Auxiliary language modeling To find a suitable value of α for combining language modeling loss and cross-entropy loss, we first observed that in the early stages of training, the former is a few orders of magnitude higher than the latter for all tasks considered. We thus selected a range $\{1\text{e}{-3}, 1\text{e}{-4}, 1\text{e}{-5}\}$ of reasonable choices for α and performed preliminary experiments on Yelp with 100 training examples to find the best value among these candidates. To this end, we split the training examples into a training set and a dev set using both a 90/10 split and a 50/50 split and took the value of α that maximizes average dev set accuracy. We adopt this value for all other tasks and training set sizes without further optimization.

Models per ensemble As we always train three models per pattern, for both iPET and training the final classifier C, the ensemble $\mathcal{M}$ (or $\mathcal{M}^0$) for n PVPs contains $3n$ models. This ensures consistency as randomly choosing any of the three models for each PVP would result in high variance. In preliminary experiments, we found this to have only little impact on the final model's performance.

iPET dataset size For iPET, we quintuple the number of training examples after each iteration ($d = 5$) so that only a small number of generations is required to reach a sufficient amount of labeled data. We did not choose a higher value because we presume that this may cause training sets for early generations to contain a prohibitively large amount of mislabeled data.

iPET dataset creation We create training sets for the next generation in iPET using 25% of the models in the current generation ($\lambda = 0.25$) because we want the training sets for all models to be diverse while at the same time, a single model should not have too much influence.

Others For all other hyperparameters listed in Table 5, we took the default settings of the Transformers library (Wolf et al., 2020).

B.2 Number of parameters

As PET does not require any additional learnable parameters, the number of parameters for both PET

and iPET is identical to the number of parameters in the underlying language model: 355M for RoBERTa (large) and 270M for XLM-R (base).

B.3 Average runtime

Training a single PET classifier for 250 steps on one GPU took approximately 30 minutes; training for 1000 steps with auxiliary language modeling took 60 minutes. Depending on the task, labeling examples from $\mathcal{D}$ took 15–30 minutes per model. Training the final classifier C for 5000 steps on the soft-labeled dataset $\mathcal{T}_C$ took 2 hours on average.

B.4 Comparison with SotA

For comparing PET to UDA (Xie et al., 2020) and MixText (Chen et al., 2020), we reduce the number of unlabeled examples by half to speed up the required backtranslation step. We use the backtranslation script provided by Chen et al. (2020) with their recommended hyperparameter values and use both Russian and German as intermediate languages.

For MixText, we use the original implementation[5] and the default set of hyperparameters. Specifically, each batch consists of 4 labeled and 8 unlabeled examples, we use layers 7, 9 and 12 for mixing, we set $T = 5$, $\alpha = 16$, and use a learning rate of $5 \cdot 10^{-6}$ for RoBERTa and $5 \cdot 10^{-4}$ for the final classification layer. We optimize the number of training steps for each task and dataset size in the range $\{1000, 2000, 3000, 4000, 5000\}$.

For UDA, we use a PyTorch-based reimplementation[6]. We use the same batch size as for MixText and the hyperparameter values recommended by Xie et al. (2020); we use an exponential schedule for training signal annealing and a learning rate of $2 \cdot 10^{-5}$. We optimize the number of training steps for each task and dataset size in the range $\{500, 1000, 1500, \ldots, 10000\}$.

B.5 In-Domain Pretraining

For in-domain pretraining experiments described in Section 5, we use the language model finetuning script of the Transformers library (Wolf et al., 2020); all hyperparameters are listed in the last column of Table 5. Pretraining was performed on a total of 3 NVIDIA GeForce GTX 1080 Ti GPUs.

[5] https://github.com/GT-SALT/MixText

[6] https://github.com/SanghunYun/UDA_pytorch

C Dataset Details

For each task and number of examples t, we create the training set $\mathcal{T}$ by collecting the first $t/|\mathcal{L}|$ examples per label from the original training set, where $|\mathcal{L}|$ is the number of labels for the task. Similarly, we construct the set $\mathcal{D}$ of unlabeled examples by selecting 10 000 examples per label and removing all labels. For evaluation, we use the official test set for all tasks except MNLI, for which we report results on the dev set; this is due to the limit of 2 submissions per 14 hours for the official MNLI test set. An overview of the number of test examples and links to downloadable versions of all used datasets can be found in Table 6.

Preprocessing In some of the datasets used, newlines are indicated through the character sequence "\n". As the vocabularies of RoBERTa and XLM-R do not feature a newline, we replace this sequence with a single space. We do not perform any other preprocessing, except shortening all examples to the maximum sequence length of 256 tokens. This is done using the *longest first* strategy implemented in the Transformers library. For PET, all input sequences are truncated *before* applying patterns.

Evaluation metrics For Yelp, AG's News, Yahoo and MNLI, we use accuracy. For x-stance, we report macro-average of F1 scores using the evaluation script of Vamvas and Sennrich (2020).

D Hyperparameter Importance

To analyze the importance of hyperparameter choices for PET's performance gains over supervised learning, we look at the influence of both the learning rate (LR) and the number of training steps on their test set accuracies.

We try values of $\{1e{-}5, 2e{-}5, 5e{-}5\}$ for the learning rate and $\{50, 100, 250, 500, 1000\}$ for the number of training steps. As this results in 30 different configurations for just one task and training set size, we only perform this analysis on Yelp with 100 examples, for which results can be seen in Figure 6. For supervised learning, the configuration used throughout the paper (LR $= 1e{-}5$, 250 steps) turns out to perform best whereas for PET, training for fewer steps consistently performs even better. Importantly, PET clearly outperforms regular supervised training regardless of the chosen learning rate and number of training steps.

Parameter	PET −LM	PET (En/Xs)	C (En/Xs)	sup. (En/Xs)	In-Dom. PT
adam_epsilon	1e-8	1e-8	1e-8	1e-8	1e-8
* alpha	–	1e-4	–	–	–
block_size	–	–	–	–	256
gradient_accumulation_steps	4	4	4	4	2
learning_rate	1e-5	1e-5	1e-5	1e-5	5e-5
max_grad_norm	1.0	1.0	1.0	1.0	1.0
max_seq_length	256	256	256	256	–
max_steps	250	1000 / –	5000 / –	250 / –	50000
mlm_probability	–	0.15	–	–	0.15
num_train_epochs	–	– / 3	– / 3	– / 3	–
per_gpu_train_batch_size	4	1	4	4	2
* per_gpu_helper_batch_size	–	3	–	–	–
* temperature	–	–	2.0	–	–
weight_decay	0.01	0.01	0.01	0.01	0.0

Table 5: Hyperparameters for training individual PET models without auxiliary language modeling (PET−LM) and with language modeling (PET), the final PET classifier (C), regular supervised training (sup.) and in-domain pretraining (In-Dom. PT). Whenever different values are used for the English datasets (En) and x-stance (Xs), both values are given separated by a slash. (*): PET-specific hyperparameters

Dataset	**Link**	**Test Examples**
AG's News	http://goo.gl/JyCnZq	7600
MNLI (m / mm)	https://cims.nyu.edu/~sbowman/multinli/	10000 / 10000
X-Stance (De / Fr / It)	https://github.com/ZurichNLP/xstance	3479 / 1284 / 1173
Yahoo! Answers	http://goo.gl/JyCnZq	60000
Yelp Review Full	http://goo.gl/JyCnZq	50000

Table 6: Download links and number of test examples for all datasets

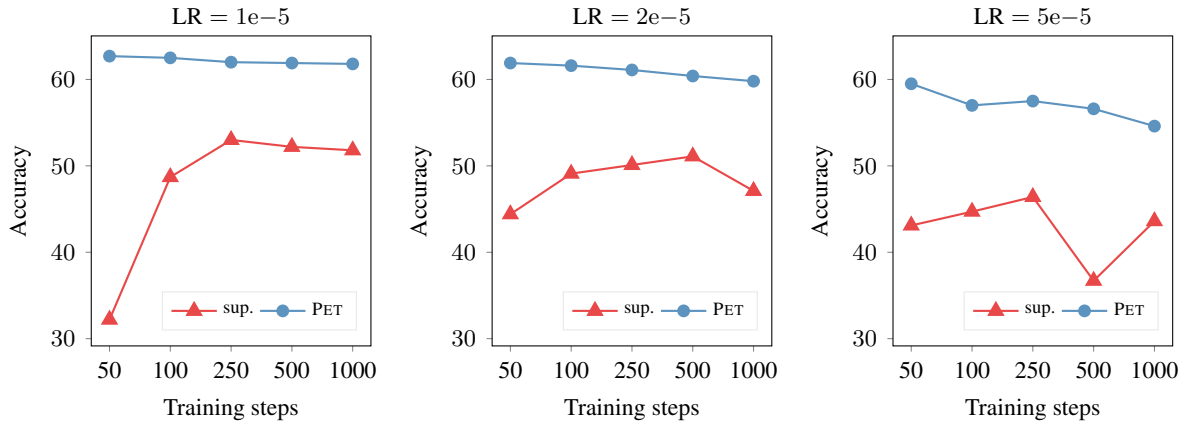


Figure 6: Performance of supervised learning and PET (weighted, without auxiliary language modeling) for various learning rates and training steps on Yelp with 100 training examples

E Automatic Verbalizer Search

Given a set of patterns $P_1, \ldots, P_n$, manually finding a verbalization $v(l)$ for each $l \in \mathcal{L}$ that represents the meaning of l well and corresponds to a single token in V can be difficult. We therefore devise *automatic verbalizer search* (AVS), a procedure that automatically finds suitable verbalizers given a training set $\mathcal{T}$ and a language model M.

Assuming we already have a PVP $\mathbf{p} = (P, v)$, we can easily check whether some token $t \in V$ is a good verbalization of $l \in \mathcal{L}$. To this end, we define $\mathbf{p}[l \leftarrow t] = (P, v')$, where v' is identical to v, except that $v'(l) = t$. Intuitively, if t represents l well, then $q_{\mathbf{p}[l \leftarrow t]}(l \mid \mathbf{x})$ (i.e., the probability M assigns to t given $P(\mathbf{x})$) should be high only for those examples $(\mathbf{x}, y) \in \mathcal{T}$ where $y = l$. We thus define the score of t for l given $\mathbf{p}$ as

$$s_l(t \mid \mathbf{p}) = \frac{1}{|\mathcal{T}_l|} \cdot \sum_{(\mathbf{x},y) \in \mathcal{T}_l} q_{\mathbf{p}[l \leftarrow t]}(l \mid \mathbf{x}) - \frac{1}{|\mathcal{T} \setminus \mathcal{T}_l|} \cdot \sum_{(\mathbf{x},y) \in \mathcal{T} \setminus \mathcal{T}_l} q_{\mathbf{p}[l \leftarrow t]}(l \mid \mathbf{x})$$

where $\mathcal{T}_l = \{(\mathbf{x}, y) \in \mathcal{T} : y = l\}$ is the set of all training examples with label l. While this allows us to easily compute the best verbalization for l as

$$\hat{t} = \arg\max_{t \in V} s_l(t \mid \mathbf{p}),$$

it requires us to already know verbalizations $v(l')$ for all other labels l'.

AVS solves this problem as follows: We first assign random verbalizations to all labels and then repeatedly recompute the best verbalization for each label. As we do not want the resulting verbalizer to depend strongly on the initial random assignment, we simply consider multiple such assignments. Specifically, we define an initial probability distribution ρ_0 where for all $t \in V, l \in \mathcal{L}$, $\rho_0(t \mid l) = 1/|V|$ is the probability of choosing t as verbalization for l. For each $l \in \mathcal{L}$, we then sample k verbalizers $v_1, \ldots, v_k$ using ρ_0 to compute

$$s_l^k(t) = \frac{1}{n \cdot k} \sum_{i=1}^{n} \sum_{j=1}^{k} s_l(t \mid (P_i, v_j))$$

for all $t \in V$.[7] These scores enable us to define a probability distribution ρ_1 that more closely reflects a word's suitability as a verbalizer for a given label:

$$\rho_1(t \mid l) = \frac{1}{Z} \max(s_l^k(t), \epsilon)$$

where $Z = \sum_{t' \in V} \max(s_l^k(t'), \epsilon)$ and $\epsilon \geq 0$ ensures that ρ_1 is a proper probability distribution. We repeat this process to obtain a sequence of probability distributions $\rho_1, \ldots, \rho_{i_{\max}}$. Finally, we choose the $m \in \mathbb{N}$ most likely tokens according to $\rho_{i_{\max}}(t \mid l)$ as verbalizers for each l. During training and inference, we compute the unnormalized score $s_{\mathbf{p}}(y \mid \mathbf{x})$ for each label by averaging over its m verbalizers.

We analyze the performance of AVS for all tasks with $|\mathcal{T}| = 50$ training examples and set $k = 250$, $\epsilon = 10^{-3}$, $i_{\max} = 5$ and $m = 10$.[8] To speed up the search, we additionally restrict our search space to tokens $t \in V$ that contain at least two alphabetic characters. Of these tokens, we only keep the 10 000 most frequent ones in $\mathcal{D}$.

Results are shown in Table 7. As can be seen, carefully handcrafted verbalizers perform much better than AVS; however, PET with AVS still considerably outperforms regular supervised training while eliminating the challenge of manually finding suitable verbalizers. Table 8 shows the most probable verbalizers found using AVS for the Yelp dataset. While most verbalizers for this dataset intuitively make sense, we found AVS to struggle with finding good verbalizers for three out of ten labels in the Yahoo dataset and for all MNLI labels.

	Yelp	AG's	Yahoo	MNLI
supervised	44.8	82.1	52.5	45.6
PET	**60.0**	**86.3**	**66.2**	**63.9**
PET + AVS	55.2	85.0	58.2	52.6

Table 7: Results for supervised learning, PET and PET with AVS (PET + AVS) after training on 50 examples

y	Top Verbalizers
1	worthless, BAD, useless, appalling
2	worse, slow, frustrating, annoying
3	edible, mixed, cute, tasty, Okay
4	marvelous, loved, love, divine, fab
5	golden, magical, marvelous, perfection

Table 8: Most probable verbalizers according to AVS for Yelp with 50 training examples

[7] Note that the score $s_l^k(t)$ jointly considers all patterns; in preliminary experiments, we found this to result in more robust verbalizers.

[8] We tried values of k and $i_{\max}$ in $\{250, 500, 1000\}$ and $\{5, 10, 20\}$, respectively, but found the resulting verbalizers to be almost identical.