Enriching Word Vectors with Sub word Information (FastText)

Presented by – Nilesh Srivastava

Contents

Introduction and Background

- FastText is a vector representation technique developed by facebook AI research.
- **fast and efficient method**
- learns **morphological** details as well.
- It can **derive word vectors for unknown words or out of vocabulary words** — this is because by taking morphological characteristics of words into account, it can *create* the word vector for an unknown word.
- **FastText works well with rare words**. So even if a word wasn't seen during training, it can be broken down into n-grams to get its embeddings.

- Earlier works ignored the internal structure of words, which is an important limitation for morphologically rich languages
- Our main contribution is to introduce an extension of the continuous skipgram model (Mikolov et al., 2013b), which takes into account subword information.
- We evaluate this model on nine languages exhibiting different morphologies, showing the benefit of our approach.

Related Works - Morphological word representations

- Alexandrescu and Kirchhoff (2006) introduced factored neural language models, where words are represented as sets of features.
- Recently, several works have proposed different composition functions to derive representations of words from morphemes (Lazaridou et al., 2013; Luong et al., 2013; Botha and Blunsom, 2014; Qiu et al., 2014).
- These different approaches rely on a morphological decomposition of words.
- Chen et al. (2015) introduced a method to jointly learn embeddings for Chinese words and characters
- Cui et al. (2015) proposed to constrain morphologically similar words to have similar representations.
- Soricut and Och (2015) described a method to learn vector representations of morphological transformations, allowing to obtain representations for unseen words by applying these rules
- Schütze (1993) learned representations of character four-grams through singular value decomposition, and derived representations for words by summing the four-grams representations. (closest approach)
- Wieting et al. (2016) also proposed to represent words using character n-gram count vectors. However, the objective function used to learn these representations is based on paraphrase pairs, while our model can be trained on any text corpus.

Related Works - Character level features for NLP

- These models discard the segmentation into words and aim at learning language representations directly from characters.
- A first class of such models are recurrent neural networks,
  - language modeling (Mikolov et al., 2012; Sutskever et al., 2011; Graves, 2013; Bojanowski et al., 2015)
  - text normalization (Chrupała, 2014)
  - part-of-speech tagging (Ling et al., 2015)
  - parsing (Ballesteros et al., 2015).
- Another family of models are convolutional neural networks trained on characters
  - part-of-speech tagging (dos Santos and Zadrozny, 2014)
  - sentiment analysis (dos Santos and Gatti, 2014)
  - text classification (Zhang et al., 2015)
  - language modeling (Kim et al., 2016).
- Sperr et al. (2013) introduced a language model based on restricted Boltzmann machines, in which words are encoded as a set of character n - grams.
- Recent works in machine translation have proposed using subword units to obtain representations of rare words (Sennrich et al., 2016; Luong and Manning, 2016).

Model - General Model (skipgram model with negative sampling)

- We model morphology by considering subword units, and representing words by a sum of its character n-grams.
- Our model is derived from continuous skipgram model introduced by Mikolov et al. (2013b)
- When a large training corpus represented as a sequence of words w1, ..., wT , the objective of the skipgram model is to maximize the following log-likelihood:

$$\sum_{t=1}^{T} \sum_{c \in \mathcal{C}_t} \log p(w_c \mid w_t),$$

Ct : set of indices of words surrounding word Wt

- let us consider that we are given a scoring function s which maps pairs of (word, context) to scores in R .

$$p(w_c \mid w_t) = \frac{e^{s(w_t, w_c)}}{\sum_{j=1}^{W} e^{s(w_t, j)}}.$$

s : Softmax function

- Above softmax function would not work for our cause as it implies that, given a word Wt , we only predict one context word Wc .
- Our goal is to independently predict the presence (or absence) of context words. (Not making it a binary classification problem)

Model - General Model (skipgram model with negative sampling)

- For the word at position t we consider all context words as positive examples and sample negatives at random from the dictionary
- For a chosen context position c, using the binary logistic loss, we obtain the following negative log-likelihood:

$$\log\left(1+e^{-s(w_t,\,w_c)}\right)+\sum_{n\in\mathcal{N}_{t,c}}\log\left(1+e^{s(w_t,\,n)}\right)$$

Nt,c is a set of negative examples sampled from the vocabulary

- By denoting the logistic loss function $l: x \to \log(1+e^{-x})$, we can re-write the objective as:

$$\sum_{t=1}^{T}\left[\sum_{c\in\mathcal{C}_t}\ell(s(w_t,\,w_c))+\sum_{n\in\mathcal{N}_{t,c}}\ell(-s(w_t,\,n))\right]$$

- Let us define for each word w in the vocabulary two vectors Uw and Vw in R^d. These two vectors are sometimes referred to as input and output vectors in the literature. In particular, we have vectors Uwt and Vwc, corresponding, respectively, to words Wt and Wc. Then the score can be computed as the scalar product between word and context vectors as $s(w_t, w_c) = \mathbf{u}_{w_t}^{\top}\mathbf{v}_{w_c}$.

Model - Subword Model

- By using a distinct vector representation for each word, the skipgram model ignores the internal structure of words
- we propose a different scoring function s, in order to take into account this information.
- Word w is represented as a bag of character n-gram, allowing to distinguish prefixes and suffixes from other character sequences.
- Taking the word where and n = 3 as an example, it will be represented by the character n-grams:

  <wh, whe, her, ere, re>
- In practice, we extract all the n-grams for n greater or equal to 3 and smaller or equal to 6.
- Suppose we have a dictionary of n - grams of size G. Given a word w, let us denote by G w $\subset$ { 1, . . . , G } the set of n-grams appearing in w. We associate a vector representation Zg to each n-gram g. We represent a word by the sum of the vector representations of its n-grams. We thus obtain the scoring function:

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c.$$

Experimental Setup

- In most experiments, we compare our model to the C implementation of the skipgram and cbow models from the word2vec 2 package.
- Optimization problem was solved by performing stochastic gradient descent on the negative log likelihood.
- In the skipgram model we use a linear decay of step size and the step size at time t is: $\gamma_0(1 - \frac{t}{TP})$, where T is number of words in training set and P is number of passes over the data.

Implementation Details:

- We use the word vectors of dimension 300.
- For each positive example, we sample 5 negatives at random, with probability proportional to the square root of the uni-gram frequency.
- A context window of size c, and uniformly sample the size c between 1 and 5. In order to subsample the most frequent words, we use a rejection threshold of 10^{-4}
- The step size γ 0 is set to 0 .025 for the skipgram baseline and to 0 .05 for both our model and the cbow baseline.
- Using this setting on English data, our model with character n-grams is approximately 1.5× slower to train than the skipgram baseline. We process 105k words/second/thread versus 145k words/second/thread for the baseline.

Experimental Setup

- These models were trained on the Wikipedia dumps for 9 languages.
  - Arabic
  - Czech
  - German
  - English
  - Spanish
  - French
  - Italian
  - Romanian
  - Russian
- We normalize the raw Wikipedia data using Matt Mahoney's pre-processing perl script. All the datasets are shuffled, and we train our models by doing five passes over them.

Results

- We evaluate our model in five experiments:
  - Human similarity judgement
  - Word analogy tasks
  - Comparison with morphological representations (comparison to state-of-the-art methods)
  - Effect of the size of the training data
  - Effect of the size of n-grams

Results : Human Similarity Judgement

- We first evaluate the quality of our representations on the task of word similarity/relatedness. It is done by computing Spearman's rank correlation coefficient between human judgement and the cosine similarity between the vector representations.
- For German, we compare the different models on three datasets: GUR65, GUR350 and ZG222.
- For English, we use the WS353 and the rare word dataset (RW).
- We evaluate the French word vectors on the translated dataset RG65.
- Spanish, Arabic and Romanian word vectors are evaluated using the datasets described in (Hassan and Mihalcea, 2009).
- Russian word vectors are evaluated using the HJ dataset introduced by Panchenko et al. (2016).
- We do have some words in these datasets that weren't present in the training set so we cannot obtain word representation for these words using the cbow and skipgram baselines.
- In order to provide comparable results, we propose by default to use null vectors for these words. Since our model exploits subword information, we can also compute valid representations for out-of-vocabulary words.

Results : Human Similarity Judgement

- We notice that the proposed model (sisg), which uses subword information, outperforms the baselines on all datasets except the English WS353 dataset.
- Computing vectors for out-of-vocabulary words (sisg) is always at least as good as not doing so (sisg-). This proves the advantage of using subword information in the form of character n-grams.
- We observe that the effect of using character n-grams is more important for Arabic, German and Russian than for English, French or Spanish.
- When evaluating on less frequent words, we see that using similarities at the character level between words can help learning good word vectors.

		sg	cbow	sisg-	sisg
AR	WS353	51	52	54	**55**
DE	GUR350	61	62	64	**70**
	GUR65	78	78	**81**	**81**
	ZG222	35	38	41	**44**
EN	RW	43	43	46	**47**
	WS353	72	**73**	71	71
ES	WS353	57	58	58	**59**
FR	RG65	70	69	**75**	**75**
RO	WS353	48	52	51	**54**
RU	HJ	59	60	60	**66**

Results : Word analogy tasks

- We now evaluate our approach on word analogy questions, of the form A is to B as C is to D , where D must be predicted by the models.
- We use the datasets introduced by Mikolov et al. (2013a) for English, by Svoboda and Brychcin (2016) for Czech, by Köper et al. (2015) for German and by Berardi et al. (2015) for Italian.
- We exclude the words that doesn't appear in the training corpus from evaluation set.

- We observe that morphological information significantly improves the syntactic tasks; our approach outperforms the baselines.
- In contrast, it does not help for semantic questions, and even degrades the performance for German and Italian.
- The improvement over the baselines is more important for morphologically rich languages, such as Czech and German.

		sg	cbow	sisg
CS	Semantic	25.7	27.6	27.5
	Syntactic	52.8	55.0	77.8
DE	Semantic	66.5	66.8	62.3
	Syntactic	44.5	45.0	56.4
EN	Semantic	78.5	78.2	77.8
	Syntactic	70.1	69.9	74.9
IT	Semantic	52.3	54.7	52.3
	Syntactic	51.5	51.8	62.7

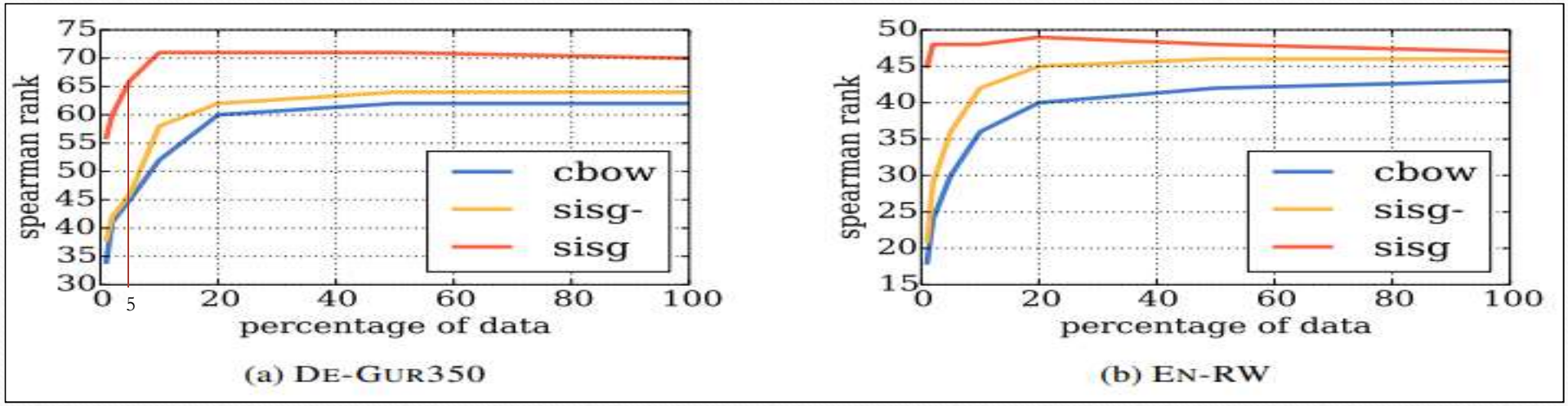
Results : Comparison with morphological representations

	DE		EN		ES	FR
	GUR350	ZG222	WS353	RW	WS353	RG65
Luong et al. (2013)	-	-	64	34	-	-
Qiu et al. (2014)	-	-	65	33	-	-
Soricut and Och (2015)	64	22	71	42	47	67
sisg	73	43	73	48	54	69
Botha and Blunsom (2014)	56	25	39	30	28	45
sisg	66	34	54	41	49	52

- The methods used are: the recursive neural network of Luong et al. (2013), the morpheme cbow of Qiu et al. (2014) and the morphological transformations of Soricut and Och (2015).
- We also compare our approach to the log-bilinear language model introduced by Botha and Blunsom (2014), which was trained on the Europarl and news commentary corpora.
- We observe that our simple approach performs well relative to techniques based on subword information obtained from morphological segmentors.
- The large improvement for German is due to the fact that their approach does not model noun compounding, contrary to ours.

Results : Effect of the size of the training data

- We train our model and the cbow baseline on portions of Wikipedia of increasing size.
- Not all words from the evaluation set are present in the Wikipedia data. So, we use a null vector for these words (sisg-) or compute a vector by summing the n-gram representations (sisg).
- The out-of-vocabulary rate is growing as the dataset shrinks, and therefore the performance of sisgand cbow necessarily degrades. However, the proposed model (sisg) assigns non-trivial vectors to previously unseen words.
- We notice that the proposed approach provides very good word vectors even when using very small training datasets.



Results : Effect of the size of n-grams

	2	3	4	5	6
2	57	64	67	69	69
3		65	68	70	70
4			70	70	**71**
5				69	**71**
6					70

(a) DE-GUR350

	2	3	4	5	6
2	59	55	56	59	60
3		60	58	60	62
4			62	62	63
5				64	64
6					**65**

(b) DE Semantic

	2	3	4	5	6
2	45	50	53	54	55
3		51	55	55	**56**
4			54	**56**	**56**
5				**56**	**56**
6					54

(c) DE Syntactic

	2	3	4	5	6
2	41	42	46	47	**48**
3		44	46	**48**	**48**
4			47	**48**	**48**
5				**48**	**48**
6					**48**

(d) EN-RW

	2	3	4	5	6
2	78	76	75	76	76
3		78	77	78	77
4			79	79	79
5				**80**	79
6					**80**

(e) EN Semantic

	2	3	4	5	6
2	70	71	73	74	73
3		72	74	**75**	74
4			74	**75**	**75**
5				74	74
6					72

(f) EN Syntactic

- We observe that taking a large range such as $3 - 6$ provides a reasonable amount of subword information.
- This is especially true for German, as many nouns are compounds made up from several units that can only be captured by longer character sequences.
- This experiment also shows that it is important to include long n-grams, as columns corresponding to $n \leq 5$ and $n \leq 6$ work best.

Results : Language Modelling

- Two baselines are considered: we compare our approach to the log-bilinear language model of Botha and Blunsom (2014) and the character aware language model of Kim et al. (2016).Our model is a recurrent neural network with 650 LSTM units, regularized with dropout (with probability of 0.5) and weight decay (regularization parameter of 10^{-5}). We learn the parameters using the Adagrad algorithm with a learning rate of 0.1, clipping the gradients which have a norm larger than 1.0. We initialize the weight of the network in the range $[-0.05, 0.05]$, and use a batch size of 20.

	CS	DE	ES	FR	RU
Vocab. size	46k	37k	27k	25k	63k
CLBL	465	296	200	225	304
CANLM	371	239	165	184	261
LSTM	366	222	157	173	262
sg	339	216	150	162	237
sisg	**312**	**206**	**145**	**159**	**206**

Text Perplexity Scores

- We report the test perplexity of our model without using pre-trained word vectors (LSTM), with word vectors pre-trained without subword information (sg) and with our vectors (sisg).
- We observe that this improvement is most significant for morphologically rich Slavic languages such as Czech (8% reduction of perplexity over sg) and Russian (13% reduction). The improvement is less significant for Roman languages such as Spanish (3% reduction) or French (2% reduction).
- CLBL refers to the work of Botha and Blunsom (2014) and CANLM refers to the work of Kim et al. (2016)

Qualitative analysis : Nearest neighbors

- We show nearest neighbors according to cosine similarity for vectors trained using the proposed approach and for the skipgram baseline.
- As expected, the nearest neighbors for complex, technical and infrequent words using our approach are better than the ones obtained using the baseline model.

query	tiling	tech-rich	english-born	micromanaging	eateries	dendritic
sisg	tile flooring	tech-dominated tech-heavy	british-born polish-born	micromanage micromanaged	restaurants eaterie	dendrite dendrites
sg	bookcases built-ins	technology-heavy .ixic	most-capped ex-scotland	defang internalise	restaurants delis	epithelial p53

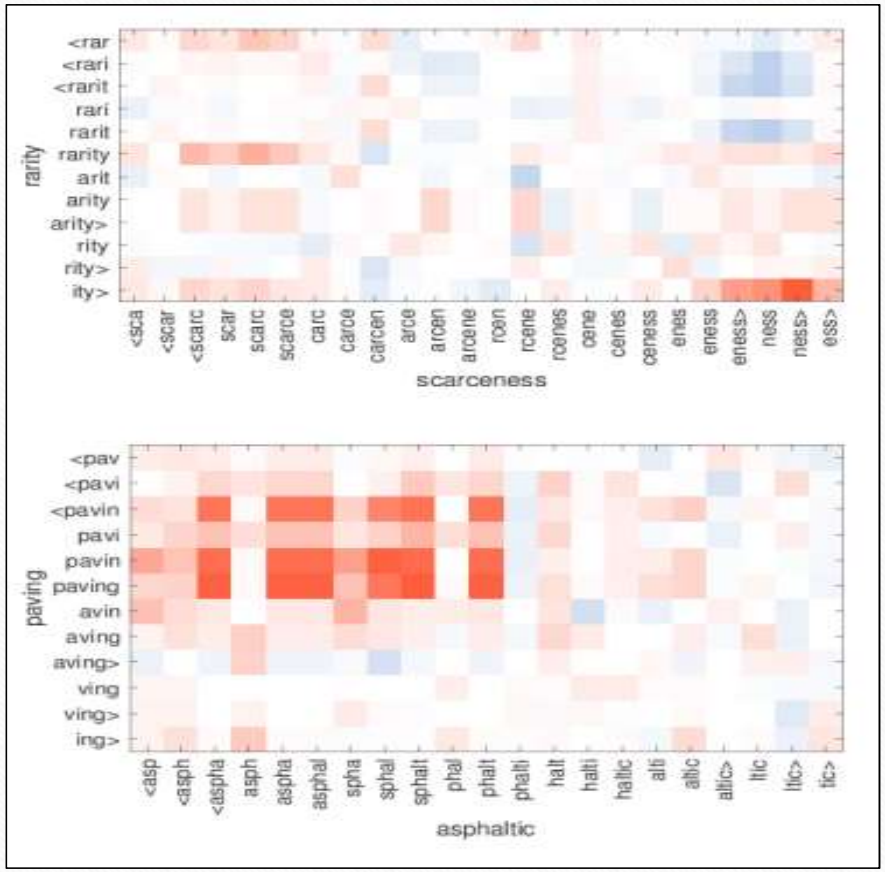
Table 7: Nearest neighbors of rare words using our representations and skipgram. These hand picked examples are for illustration.

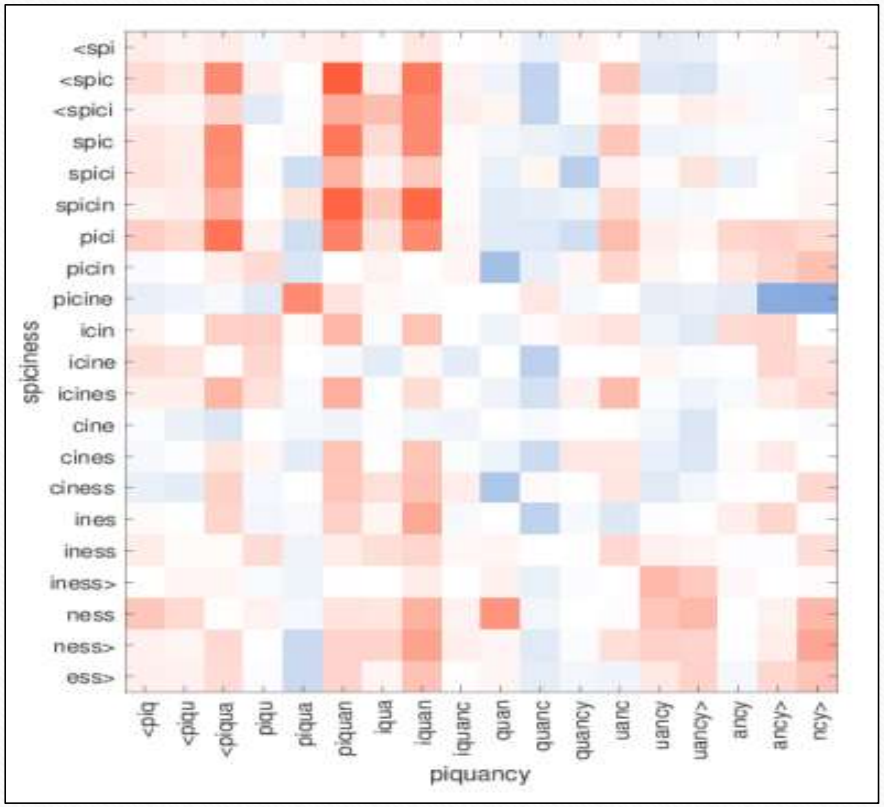
Qualitative analysis : Character n-grams and morphemes

- We want to qualitatively evaluate whether or not the most important n-grams in a word correspond to morphemes.
- For German, which has a lot of compound nouns, we observe that the most important n-grams correspond to valid morphemes.
- Good examples include Autofahrer (car driver) whose most important n-grams are Auto (car) and Fahrer (driver).
- We also observe the separation of compound nouns into morphemes in English, with words such as lifetime or starfish. However, for English, we also observe that n-grams can correspond to affixes in words such as kindness or unlucky.
- For French we observe the inflections of verbs with endings such as ais> , ent> or ions> .

	word	*n*-grams		
DE	autofahrer	fahr	fahrer	auto
	freundeskreis	kreis	kreis>	<freun
	grundwort	wort	wort>	grund
	sprachschule	schul	hschul	sprach
	tageslicht	licht	gesl	tages
EN	anarchy	chy	<anar	narchy
	monarchy	monarc	chy	<monar
	kindness	ness>	ness	kind
	politeness	polite	ness>	eness>
	unlucky	<un	cky>	nlucky
	lifetime	life	<life	time
	starfish	fish	fish>	star
	submarine	marine	sub	marin
	transform	trans	<trans	form
FR	finirais	ais>	nir	fini
	finissent	ent>	finiss	<finis
	finissions	ions>	finiss	sions>

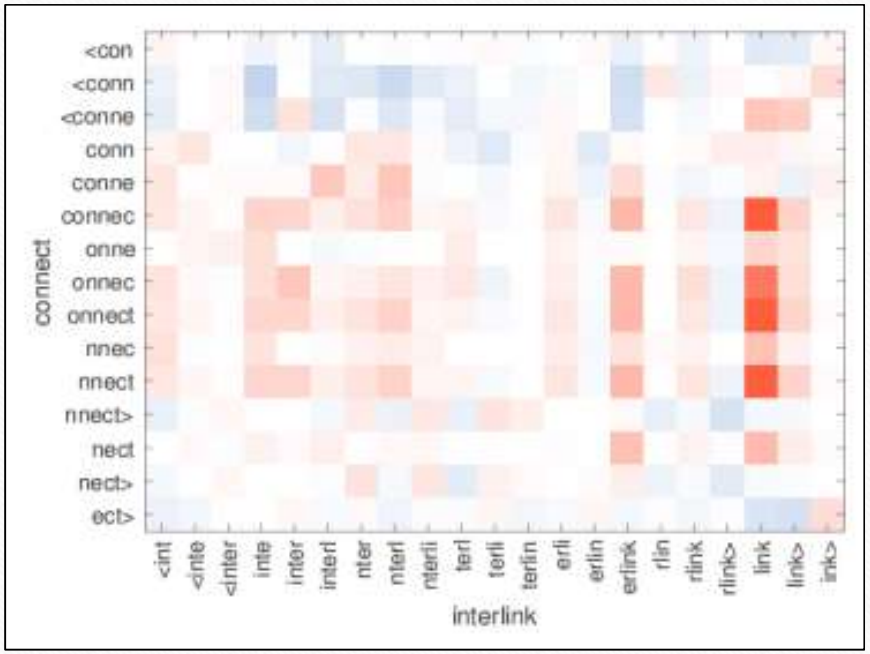
Qualitative analysis : Word similarity for OOV words

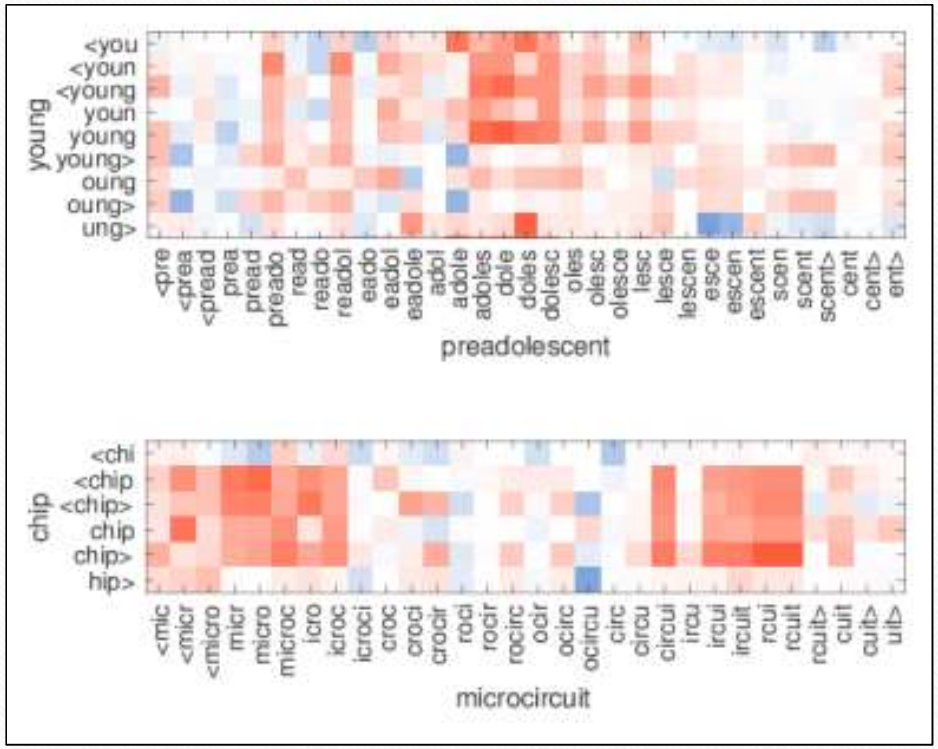




Red indicates positive cosine, while blue negative.

Qualitative analysis : Word similarity for OOV words





Conclusion

- In this review, we investigated a simple method to learn word representations by taking into account subword information.
- Our approach, which incorporates character n-grams into the skipgram model, is related to an idea that was introduced by Schütze (1993).
- Because of its simplicity, our model trains fast and does not require any preprocessing or supervision.
- We show that our model outperforms baselines that do not take into account subword information, as well as methods relying on morphological analysis.

References

- Enriching Word Vectors with Subword Information by Piotr Bojanowski* and Edouard Grave* and Armand Joulin and Tomas Mikolov (Facebook AI Research)
- https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00051/43387/Enriching-Word-Vectors-with-Subword-Information