# Supersense Tagging with a Combination of Character, Subword, and Word-level Representations

**Youhyun Shin** and **Sang-goo Lee**
Department of Computer Science and Engineering
Seoul National University
shinu89,sglee@europa.snu.ac.kr

## Abstract

Recently, there has been increased interest in utilizing characters or subwords for natural language processing (NLP) tasks. However, the effect of utilizing character, subword, and word-level information simultaneously has not been examined so far. In this paper, we propose a model to leverage various levels of input features to improve on the performance of an supersense tagging task. Detailed analysis of experimental results show that different levels of input representation offer distinct characteristics that explain performance discrepancy among different tasks.

## 1 Introduction

Recently, there has been increased interest in using characters or subwords, instead of words, as the basic unit of language feature in natural language processing tasks. Utilizing subword information has been shown to be very effective for named entity alignment of parallel corpus (Sennrich and Haddow, 2016) and named entity recognition (Lample et al., 2016; Santos and Guimaraes, 2015). Some recent advancements were achieved using character or subword features in neural machine translation and language modeling (Sennrich et al., 2015; Chung et al., 2016; Lee et al., 2016; Kim et al., 2016).

The main benefit of utilizing features below word-level is the ability to overcome out-of-vocabulary (OOV) and the rare word problems. When faced with very infrequent or OOV words in the test data, word-level models must resort to replacing them with "unknown word" tokens; and in many cases, this discarded information could be vital for understanding certain semantics of the text, hence word-level models could perform poorly when said types of words appear frequently.

Traditionally, words are segmented into subwords using carefully engineered morpheme analyzers (Smit et al., 2014). Recently, we see a rise in popularity of data-driven methods such as employing an efficient encoding scheme of character sequences (e.g. byte-pair encoding (Sennrich et al., 2016)). Words could also be split into individual characters to capture even finer syntactic details. Subword schemes of varying linguistic granularity offer a trade-off between capturing semantic and syntactic features.

Despite of the success of character or subword-level approaches, there has been lack of studies on ways to combine different levels of features, namely character, subword, and word-level features. To the best of our knowledge, utilization of subword units have not even been applied to supersense tagging yet. In this paper, we present a novel neural network architecture that incorporates all three types of word-feature units (Section 3). We conduct experiments on SemCor dataset using our model (Section 4.2). Then we analyze the optimal combination of the word features for each classs of the 41 supersenses in detail (Section 4.3).

## 2 Background

### 2.1 Supersense Tagset

The supersense tagset consists of a total of 41 supersenses which are top-level semantic classes used in WordNet (Fellbaum, 1998) as shown in Table 1. This set is generally used for evaluating the approaches to coarse-grained word sense disambiguation and information extraction such as extended NER (Ciaramita and Johnson, 2003; Ciaramita and Altun, 2006). In this paper, we use the SemCor dataset (Table 2) for evaluation.

| Nouns | | | | | | | |
|---|---|---|---|---|---|---|---|
| Supersense | Freq. | Supersense | Freq. | Supersense | Freq. | Supersense | Freq. |
| person | 17% | attribute | 5% | object | 2% | process | 1% |
| artifact | 10% | time | 5% | possession | 2% | plant | 1% |
| act | 9% | state | 4% | phenomenon | 1% | shape | <1% |
| cognition | 8% | body | 3% | animal | 1% | motive | <1% |
| group | 7% | substance | 2% | relation | 1% | Tops | <1% |
| communication | 8% | quantity | 2% | feeling | 1% | | |
| location | 5% | event | 2% | food | 1% | | |
| Verbs | | | | | | | |
| stative | 26% | social | 7% | perception | 5% | body | 2% |
| communication | 13% | motion | 8% | creation | 4% | competition | <1% |
| change | 9% | possession | 6% | emotion | 2% | weather | <1% |
| cognition | 9% | contact | 6% | consumption | 2% | | |

Table 1: The 41 WordNet supersenses (26 nouns and 15 verbs) and their frequency percentages. Note the sum of the percentages of the nouns is 100%, and that of the verbs is 100%.

| | Train | Test | Total |
|---|---|---|---|
| Documents | 150 | 36 | 186 |
| Sentences | 15,462 | 4,676 | 20,138 |
| Subwords | 436,101 | 104,264 | 540,365 |
| Words | 348,987 | 85,787 | 434,774 |
| Supersenses | 109,183 | 25,952 | 135,135 |
| Nouns | 71,919 | 15,506 | 87,425 |
| Verbs | 37,264 | 10,446 | 47,710 |

Table 2: The statistics of the SemCor dataset.

### 2.2 Subword Segmentation

We use Byte Pair Encoding (BPE) (Sennrich et al., 2016) to segment words into subwords. First, BPE produces the most efficient character encoding scheme given a corpus. The encoding scheme consists of a fixed-size dictionary containing the most frequent character sequences. If a word is not frequent enough to be listed in the dictionary, it is broken down into subwords that exist in the dictionary and the meaning of the word is inferred from the meanings of the subwords. For example, an infrequent word "transition" could be split into frequent character sequences "transi@@" and "tion".

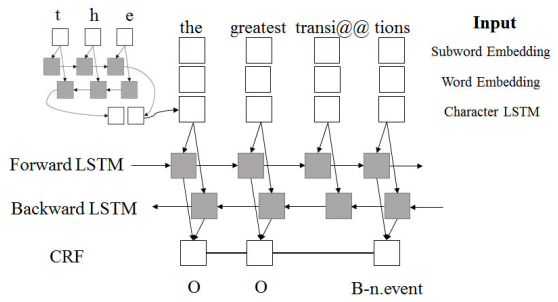


Figure 1: Model architecture

## 3 Model Description

We define *supersense tagging* as a sequence labeling problem: given an input word sequence $W = (w_1, w_2, \ldots, w_n)$, it is segmented into subword sequences using some encoding scheme (e.g. BPE) $X = (x_1, x_2, \ldots, x_m)$, and then a sequence of supersense labels $Y = (y_1, y_2, \ldots, y_n)$ is predicted, where $y \in \{1, \ldots, k\}$.

We present a novel neural network model that incorporates all of varying levels of word features: character, subword, and word (Figure 1). This model is similar to (Lample et al., 2016), but differs from it in that (i) our model uses subword-level features as the basic unit of the main LSTM architecture (Section 3.2), (ii) uses delayed prediction to synchronize subword-level sequences with word-level predictions (Section 3.2), and (iii) takes subword-level input representations along with characters and words (Section 3.1).

### 3.1 Input Representation

For each $x$, our model produces three types of embeddings: (i) character-level embedding $\mathbf{z}^{(c)}$, (ii) subword embedding $\mathbf{z}^{(s)}$, and (iii) word-level embedding $\mathbf{z}^{(w)}$. In order to produce character-level representation, a bidirectional long short-term memory cell (LSTM) $\mathrm{BiLSTM_c}$ is utilized. The hidden states of either directions of the cells are concatenated into a single character-level representation: $\mathbf{c} = \left[\mathbf{h}^{(f)}; \mathbf{h}^{(b)}\right]$. Producing subword embeddings is trivial, as each $x$ is assigned a trainable vector $\mathbf{z}^{(s)}$. Lastly, a word embedding $\mathbf{z}^{(w)}$ is produced by taking the embedding of the word in which $x$ belongs. Note for some experiments, we

use Glove to initialize word embeddings [1]. These representations are concatenated to produce a single vector $\mathbf{z} \in \mathbb{R}^r$ for each $x$, where $r$ is the subword embedding dimension:

$$\mathbf{z} = \left[\mathbf{z}^{(c)}; \mathbf{z}^{(s)}; \mathbf{z}^{(w)}\right] \quad (1)$$

### 3.2 BiLSTM-CRF Architecture

We employ BiLSTM-CRF as the base architecture. Unlike previous work, subword-level embeddings instead of word-level embeddings are fed in at each time step. Given a subword-level embedding sequence $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m)$, the main bidirectional LSTM, $\mathrm{BiLSTM_s}$, along with a synchronization layer $L_s$, and a linear layer $L_o$, produce prediction scores $\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_n)$.

$$\mathbf{H}^{(s)} = \mathrm{BiLSTM_s}\left(\mathbf{Z}\right) \quad (2)$$

$$\mathbf{H}^{(w)} = L_s\left(\mathbf{H}^{(s)}\right) \quad (3)$$

$$\mathbf{O} = L_o\left(\mathbf{H}^{(w)}\right) \quad (4)$$

Note that due to the difference between input and output lengths $m$ and $n$, synchronization between the two adjacent layer is required. The synchronization layer delays the supersense prediction until a word is fully formed by its subwords. Untrainable layer $L_s$ is implemented by selectively allowing hidden outputs, where the subword aligns with the ending of the word it belongs to, pass through the layer:

$$\mathbf{H}^{(w)} = \mathbf{W}^{(s)} \cdot \mathbf{H}^{(s)} \quad (5)$$

Where $\mathbf{W}^{(s)} \in \mathbb{R}^{n \times m}$ and each element is defined as $\mathbf{W}^{(s)}_{i,j} = \mathbb{1}\left(\mathrm{end}\left(x_j, w_i\right)\right)$. Then the output layer $L_o$ applies linear transformation on $\mathbf{H}^{(w)}$ to produce label scores $\mathbf{O} \in \mathbb{R}^{n \times k}$.

As the final layer, the conditional random field (CRF) takes time-independent label scores $\mathbf{O}$ and produces a joint score of the entire sequence by considering interdependency among labels:

$$s\left(W, Y\right) = \sum_{i=0}^{n} \mathbf{A}_{y_i, y_{i+1}} + \sum_{i=1}^{n} \mathbf{O}_{i, y_i} \quad (6)$$

Where $\mathbf{A}$ is the transition matrix among labels. For all $Y$, we maximize

$$\log p\left(Y|W\right) = s\left(W, Y\right) - \log \sum_{\bar{Y}} e^{s\left(W, \bar{Y}\right)} \quad (7)$$

Where $\bar{Y}$ is all possible combinations. Maximizing the objective encourges the valid sequence of labels to be produced.

## 4 Experiments

### 4.1 Experimental Setup

Dropout rate was 0.5, stochastic gradient descent (SGD) was used as learning method, and learning rate was 0.005. The gradient clipping is 5.0.

### 4.2 SemCor Evaluations[2]

| without pre-trained vectors | | | |
|---|---|---|---|
| | Precision | Recall | F-score |
| char | 51.4 | 48.7 | 50.02 |
| sub | 63.5 | 63.6 | 63.54 |
| word | 64.9 | 63.6 | 64.30 |
| s+w | 64.1 | 62.0 | 63.04 |
| c+s | 65.1 | 65.9 | 65.46 |
| c+w | 66.8 | 66.3 | 66.51 |
| c+s+w | 64.0 | 65.0 | 64.47 |
| with pre-trained word vector | | | |
| word | 66.9 | 67.5 | 67.20 |
| s+w | 68.0 | 68.4 | 68.20 |
| c+w | 68.6 | 69.5 | 69.04 |
| c+s+w | 68.1 | 69.5 | 68.82 |
| with pre-trained subword & word vectors | | | |
| c+s+w | 68.9 | 69.7 | 69.32 |

Table 3: Comparison of character, subword, and word-level models with/without pre-trained vectors.

The classification results of SemCor dataset using different combinations of input representations are shown in Table 3. We note that in uni-representation settings the word-level model performs better than the character or subword-level model. This is presumably because supersense tagging predicts labels for each word. We also note that when word embeddings are pre-trained, the performance is always improved by the addition of character or subword-level embeddings. Overall, the best result is obtained when the subword and word embeddings are pre-trained and all embeddings are utilized.

### 4.3 Detailed Analysis

To investigate the effect of using character or subword-level embeddings, we select 15 supersenses and examine each of them individually (Table 4). With pre-trained vectors, *c+s+w* performs much better than other combinations, outperforming others in many classes. However, without the

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

[2] We use shorthands *c*, *s* and *w* to denote *character*, *subword* and *word*, respectively.

| | without pre-trained vectors | | | | | | | with pre-trained vectors | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | c | s | w | c+s | c+w | s+w | c+s+w | w | c+w | s+w | c+s+w | c+s+w |
| Named Entity Recognition supersenses | | | | | | | | | | | | |
| person | 77.8 | 75.6 | 72.7 | **84.7** | 84.6 | 70.2 | 82.2 | 86.4 | 90.6 | 86.1 | 89.3 | **90.7** |
| group | 50.8 | 58.7 | 60.4 | **64.4** | 63.7 | 61.0 | 63.6 | 62.5 | 65.1 | 62.0 | 64.2 | **65.6** |
| location | 46.1 | 56.3 | 57.6 | 60.3 | **60.9** | 51.4 | 59.9 | 61.0 | 69.5 | 63.7 | 67.7 | **70.2** |
| 3 most frequent noun and verb supersenses | | | | | | | | | | | | |
| artifact | 50.2 | 65.8 | 67.5 | 67.0 | 67.6 | 67.5 | **67.9** | 70.8 | **74.8** | 74.4 | 73.8 | 73.6 |
| act | 40.4 | 55.8 | 58.4 | 58.8 | **60.3** | 56.6 | 57.9 | 58.9 | 62.6 | 60.7 | **63.3** | 62.5 |
| cognition | 41.7 | 59.2 | 59.2 | **60.6** | 58.1 | 56.2 | 60.1 | 59.8 | 59.8 | 60.8 | 61.2 | **61.6** |
| stative | 72.0 | 76.3 | 75.9 | 77.0 | **77.7** | 75.2 | 75.6 | 76.1 | 77.4 | 77.3 | **78.6** | 78.2 |
| communication | 55.1 | 73.7 | **76.4** | 75.4 | 72.7 | 76.3 | 73.5 | 75.9 | 76.6 | 76.4 | **78.8** | 78.3 |
| change | 32.9 | 54.4 | 52.9 | 54.7 | **55.9** | 54.4 | 54.5 | 56.2 | **59.9** | 56.5 | 58.6 | 59.5 |
| 3 rarest noun and verb supersenses | | | | | | | | | | | | |
| shape | 0.0 | 16.7 | 28.6 | 17.9 | 28.2 | **30.2** | 26.5 | 23.3 | 31.4 | 26.8 | 22.2 | **32.9** |
| motive | 57.6 | 75.0 | 73.9 | 68.8 | 76.3 | **84.2** | 75.9 | **76.9** | 69.1 | 75.0 | 71.8 | 76.6 |
| Tops | 71.4 | 80.0 | **85.7** | **85.7** | 66.7 | **85.7** | 75.0 | **85.7** | 71.4 | 76.9 | 80.0 | 75.0 |
| body | 8.5 | 43.3 | **49.7** | 40.4 | 45.7 | 43.8 | 44.1 | 50.0 | 49.9 | 49.4 | 47.9 | **51.1** |
| competition | 0.0 | 34.7 | 32.9 | 35.5 | **37.0** | 32.6 | 34.6 | 34.6 | 36.3 | 34.3 | 36.7 | **39.0** |
| weather | 0.0 | 0.0 | **27.3** | 10.5 | 19.1 | 0.0 | 17.4 | 0.0 | 0.0 | 9.5 | **22.2** | 19.1 |

Table 4: F-score comparison for NER, the most frequent and rarest supersense classification. Bold values are best cases in with/without pre-trained vectors, respectively. Underlined values represent the cases that use pre-trained embeddings.

| | char | sub | word | c+s | c+w | s+w | c+s+w |
|---|---|---|---|---|---|---|---|
| **Mr.** | Griston | Thomas | Bob | Dr. | Pope | William | Mrs. |
| | Ledford | Bob | Dr. | Mrs. | Vice | Bob | Dr. |
| | Jasper | Sam | Alexander | Sen. | Mollie | Dr. | Bob |
| **States** | Swedes | states | states | State | Paris | states | Angeles |
| | Pisces | cities | State | Moscow | State | places | heaven |
| | Seldes | S. | state | Lewis | Spots | Manchester | outside |
| **with** | wither | With | With | from | With | With | With |
| | With | by | o'clock | With | from | on | possible |
| | within | By | breakin | behind | without | On | On |

Table 5: Nearest neighbors analysis based on different representation vectors.

pre-trained vectors, it fails to maintain the dominance.

Also, 5 out of the 7 combinations perform better than others in at least one class. This shows that character, subword, and word-level embeddings offer features of different characteristics that could be either advantageous or disadvantageous depending on the class.

We further conduct nearest neighbor analysis on various embedding combinations (Table 5). We find that, in most cases, words of the same supersenses are mapped closely to each other in the word embedding space. Similar to our findings in previous analysis, we also find that each model exhibits distinct characteristics. For example, in *c+s* model, the nearest neighbors of *Mr.* are *Dr.* and *Mrs.*. However, in *sub* model, male names such as *Thomas* and *Bob* are identified as the nearest neighbors.

## 5 Conclusion

In this paper, we examine the effect of various combinations of input representations on the performance of supersense tagging task. Furthermore, a modified BiLSTM-CRF model which is able take subword sequences and predict word labels is proposed. Our experiments on supersense tagging show that utilizing all token units (character, subword, and word-level) along with pre-trained word vectors perform the best. Based on detailed analysis of selective supersense classes, we conjecture that each granlarity level of input representations offers different semantic and syntactic features that could have varying effects depending on the task. As future work, we intend to investigate the feasibility of a model that self-learns the optimal continuous combination of different levels of subword information depending on the task and data characteristics.

## References


Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 594–602.

Massimiliano Ciaramita and Mark Johnson. 2003. Supersense tagging of unknown nouns in wordnet. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*. Association for Computational Linguistics, pages 168–175.

Christiane Fellbaum. 1998. Wordnet: An electronic lexical database.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*. pages 260–270.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017* .

Cicero Nogueira dos Santos and Victor Guimaraes. 2015. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008* .

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, Mikko Kurimo, et al. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *The 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Gothenburg, Sweden, April 26-30, 2014*. Aalto University.