Artificial Intelligence Review (2023) 56:5645–5728
https://doi.org/10.1007/s10462-022-10300-7

A survey on syntactic processing techniques

Xulang Zhang[1] · Rui Mao[1] · Erik Cambria[1]

Published online: 8 November 2022

Abstract
Computational syntactic processing is a fundamental technique in natural language processing. It normally serves as a pre-processing method to transform natural language into structured and normalized texts, yielding syntactic features for downstream task learning. In this work, we propose a systematic survey of low-level syntactic processing techniques, namely: microtext normalization, sentence boundary disambiguation, part-of-speech tagging, text chunking, and lemmatization. We summarize and categorize widely used methods in the aforementioned syntactic analysis tasks, investigate the challenges, and yield possible research directions to overcome the challenges in future work.

Keywords Microtext normalization · Sentence boundary disambiguation · Part-of-speech tagging · Text chunking · Lemmatization · Syntactic processing · Natural language processing

1 Introduction

Syntactic processing is a generalization of natural language processing (NLP) subtasks that are concerned with the structure of phrases and sentences, as well as the relation of words to each other within the phrase or sentence (Woolf 2009). It involves extracting meanings from sentence constituents, and establishing the semantic structure of the input sentence (Roberts 2016). Syntactic processing is thus a foundational step towards higher level of interpretations, preceding other more complicated tasks, such as word sense disambiguation, information retrieval, sentiment analysis, dialogue system, etc. It pre-processes the input texts, and provides important features that can be further utilized by machines. However, in the era of deep learning based high-level NLP tasks, syntactic processing techniques are often overlooked. These NLP applications rely on neural networks

✉ Erik Cambria
cambria@ntu.edu.sg

Xulang Zhang
xulang001@e.ntu.edu.sg

Rui Mao
rui.mao@ntu.edu.sg

1 School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore

to implicitly extract syntactic information. They are indeed able to significantly outperform symbolic methods that depend on handcrafted feature templates. Nevertheless, we argue that integrating symbolic and subsymbolic Artificial Intelligence (AI), also known as neurosymbolic AI, is the key for stepping forward in the path from NLP to natural language understanding (Cambria et al. 2017). Despite of the recent advancement of deep learning, it does not truly achieve human-like natural language understanding, as they simply make probabilistic guesses. Belinkov et al. (2018) conducted an experiment to evaluate the word embeddings learned by Neural Machine Translation (NMT) on syntactic tasks, with results indicating that they are poor representations for syntactic information. Therefore, to further improve the performance of semantic and pragmatic tasks, it is of great benefit to regard syntactic processing as sub-problems. For instance, Part-of-Speech (POS) tagging is incorporated in the decoding process (Feng et al. 2019) or as an auxiliary task (Niehues and Cho 2017; Mao and Li 2021) to improve NMT and metaphor detection, lemmatization as a pre-processing step boosts the accuracy of neural sentiment analysis on social media text (Symeonidis et al. 2018). Hence, syntactic processing is an integral part of the neurosymbolic NLP paradigm.

In this paper, we examine the five most basic tasks in syntactic processing, namely, microtext normalization, Sentence Boundary Disambiguation (SBD), POS tagging, text chunking, and lemmatization. There is a variety of other tasks in this field, e.g., stop word removal, negation detection, constituency parsing, and dependency parsing. Here, we decide to focus on the most foundational syntactic processing tasks that can be helpful to high-level syntactic tasks and other NLP applications. To elaborate, dependency parsing analyzes the grammatical structure of a sentence based on the dependencies between words. It is strongly related to POS linguistically. Indeed the majority of existing algorithms heavily rely on POS tagging (Chen and Manning 2014; Dyer et al. 2015; Dozat and Manning 2016; Zhou et al. 2020). Similarly, constituency parsing analyzes the sentences by breaking down the sentences into constituents. It can be regarded as hierarchical text chunking, establishing structure within the chunks. Thus, we consider them to be high-level tasks that can benefit from the introduced low-level syntactic processing techniques and warrant reviews on their own.

Although low-level syntactic processing tasks have gradually faded out of public views in many NLP conference proceedings, previous surveys presented significant findings in individual syntactic processing tasks. Satapathy et al. (2020) presented a comprehensive review on microtext normalization. They discussed the similarities between microtext and brachygraphy, and suggested the potential applications of microtext normalization in more complex NLP tasks. Read et al. (2012) reviewed 10 publicly available SBD systems, and argued that the published results of these systems are evaluated on different task definitions and data annotations. Thus, they assessed the systems on unified task definition and gold-standard datasets, as well as on user generated content corpora. Manning (2011) conducted error analysis on the existing POS tagging algorithms at the time, deducing 7 common error categories, and proposed a solution for the errors and inconsistencies in the gold-standard dataset. Kanis and Skorkovská (2010) compared manual and automatic Czech lemmatizers, based on their influence on the performance of information retrieval. However, many of these reviews are out-of-date, unable to cover the recent advancement, powered by e.g., deep learning techniques. Additionally, some of them focused on the applications of specific tasks. In this work, we aim to provide an extensive, up-to-date survey for the low-level syntactic tasks in a package, inspiring novel applications in the emerging neurosymbolic AI paradigm, bringing the low-level syntactic tasks back to the vision of the researchers of high-level NLP tasks.

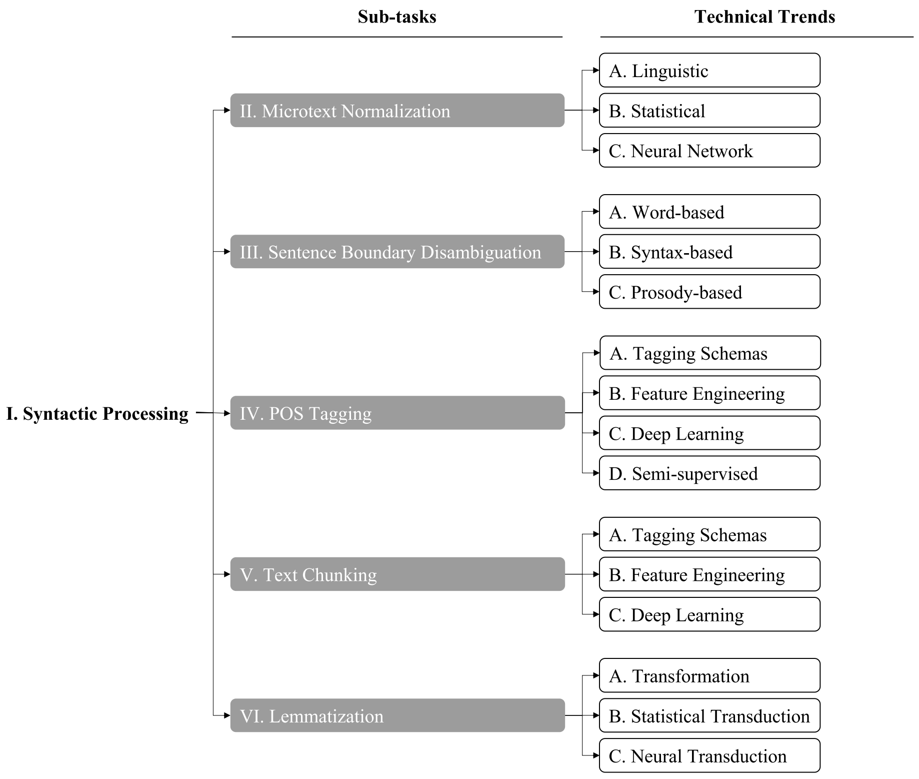

Fig. 1 The outline of the survey. By the end of each sub-task, we highlight the features of different technical trends

We introduce the tasks in the order of microtext normalization, SBD, POS tagging, text chunking, and lemmatization. For each of the tasks, we (a) give an overview of the problem; (b) categorizing and reviewing the existing algorithms; and (c) delivering a summary about the features of different technical trends, challenges to be addressed, and possible paths and directions in future work. A more detailed outline of the taxonomies in this survey is illustrated in Fig. 1. In closing, we briefly list some examples of applying syntactic processing techniques to complex NLP tasks and conclude our survey.

2 Microtext normalization

In the era of Short Message Service (SMS) and social media, a plethora of text data can be mined from the web. Microtext is a term referring to shorthand writing, commonly seen in these informal texts. The occurrence of microtext poses a problem for NLP, as it uses informal languages, e.g., shortened and lengthened words, abbreviated phrases, and missing grammatical components. It is difficult for machines to understand the meaning of the source text. Hence, microtext normalization is crucial in NLP. The task aims to convert the informal texts into their standard forms that can be easily processed by a downstream task, such as sentiment analysis for social media (Brody and Diakopoulos 2011; Satapathy et al. 2017, 2019b), information retrieval (Bontcheva et al. 2013), and question answering system (Mittal et al. 2014).

Microtext is often highly compact, informal, and semi-structured (Rosa and Ellen 2009). There are many abbreviations, acronyms, lax spelling and grammar, as well as metadata that establish the context. Although microtext is a written language, it is strongly influenced by phonetics, resulting in language-dependent character repetition and substitution phenomena (Satapathy et al. 2020), e.g., *"nooooo"* for *"no"*, and *"luv"* for *"love"*. Even without any knowledge about texting conventions, humans can understand this type of irregular words by sounding them out. Thus, phonetics-based abbreviations and spelling alterations are invented and evolve more freely than orthography-based ones. For instance, *"schewpid"* is created to imitate *"stupid"* in a British accent. It is uncommonly used, but most English speakers can understand it, even if they have never seen it before. As for orthography-based abbreviations and alterations such as *"wdym"* for *"what do you mean"*, less Internet-savvy people will have to look up its meaning. Therefore, text-level mapping alone is often not adequate to address the sparsity caused by phonetic-level alterations. Thus, existing typical corpora (see Table 1) are prepared for both text-level and phonetic-level mappings. Accuracy, top-*n* accuracy, F-score, and BLEU score (Papineni et al. 2002) are widely used evaluation measures. Specifically, top-*n* accuracy considers the system correct, if the correct standard form is in its top *n* predictions.

We categorize the existing microtext normalization approaches by linguistic, statistical, and neural network methods.

2.1 Linguistic approach

Linguistic approaches normalize microtext by modeling linguistic knowledge or commonly observed patterns in informal texts. Normalizers in this categories use rule-based and/or lexicon-based method to leverage orthographic and phonetic information, or to implement their observations or assumptions about the formation of microtext. This approach is advantageous at addressing specific problems, such as abbreviation, character repetition, and character substitution caused by phonetics. However, it is labor intensive, as most rules and lexicons are manually built.

Pennell and Liu (2010) described a rule-based normalization system for text messages so that they can be read by a text-to-speech (TTS) system. A set of hand-crafted rules are applied to translate standard English into abbreviations in text lingo, which provides features for a Maximum Entropy (MaxEnt) classifier to automatically create abbreviations, thereby establishing a mapping from standard English to the text message domain. This mapping is then reversed to create a look-up table so that the corresponding English words can be predicted from the input text message.

Brody and Diakopoulos (2011) described an automated method based on the notion that word lengthening is strongly associated with subjectivity and sentiment. Their method leverages this association to identify new domain-specific sentiment- and emotion-bearing words that are not in the existing lexicon. A rule-based unsupervised procedure is applied to detect such OOV words, generate the possible canonical forms, and discover their polarity.

Jose and Raj (2014) proposed a lexico-syntactic normalization model. As the name suggests, their model consists of two modules - lexical normalization module, and syntactic analysis module. The former is based on a channelized database and a user feedback system. Inspired by the multi-channel model (Xue et al. 2011) that is introduced in the later

Table 1 The text-level and phonetic-level corpora for microtext normalization

Category	Dataset	Source	Reference
Text-level	SMS-M	Mobile phone short messages	Dunlop and Crossan (2000)
	SMS-C	Short Message Service corpus	Choudhury et al. (2007)
	SMS-L	Mobile phone short messages	Liu et al. (2011)
	NUS SMS	NUS Short Message Service corpus	How and yen Kan (2005) and Wang and Ng (2013)
	Edinburgh	The Edinburgh Twitter Corpus	Petrović et al. (2010)
	LexNorm1.1	English Twitter text	Han and Baldwin (2011a)
	LexNorm1.2	Modified LexNorm1.1	Yang and Eisenstein (2013)
	W-NUT	2015 ACL-IJCNLP Workshop on Noisy User-generated Text	Baldwin et al. (2015)
	Nom-Tweets	English Twitter text	Satapathy et al. (2017)
	WSJ	Wall Street Journal	Marcus et al. (1993) and Wilcox-O'Hearn et al. (2008)
	WUS	SMS messages in Swiss	Ueberwasser and Stark (2017)
	Sw-SMS[a]	SMS messages in Swiss	Lusetti et al. (2018)
Phonetic-level	ARPABET	International Phonetic Alphabet (IPA) to ASCII symbols	Khoury (2015)
	TIMIT	Revised ARPABET	Taylor et al. (1998)
	CMUdict[b]	Carnegie Mellon University Pronouncing Dictionary	Jiampojamarn et al. (2007)
	Festival	Festival lexicon	Taylor et al. (1998)
	CELEX[c]	Lexical databases of English, Dutch and German	Jiampojamarn et al. (2007)

[a]https://sms.linguistik.uzh.ch/

[b]http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[c]http://celex.mpi.nl/

section, the database is built as four channels of database, namely abbreviation channel, non-noisy channel, grapheme channel, and phoneme channel. Given a tokenized input string, candidate outputs are produced from each of these channels via binary search. The syntactic analysis module is based on a bottom-up parser. It takes the lexically normalized string outputted from the previous module, and builds the corresponding parsing tree using grammar rules. The module will then decide whether the sentence is valid or not, which helps to make the text normalization task more efficient.

Desai and Narvekar (2015) introduced a method to normalize OOV words. First, a set of rules are applied to standardize elongated words such as "gooooood". Then, the most probable candidates are found from compiled databases for the OOV tokens. Lastly, any remaining noises are normalized using Levenshtein edit distance, which signifies the minimum number of single character insertions, deletions, and substitutions required to transform one word to another.

Mittal et al. (2014) proposed a three-level architecture to process microtext in Question Answering (QA) system. First, noise presented in the SMS tokens are replaced with the closest dictionary words using the Soundex (Beider 2008), Metaphone algorithm (Philips 1990), and a modified version of the Longest Common Subsequence (LCS) algorithm termed Phonetic LSC (PLSC) that also takes phonetic similarity into account. Next, a semantically similar set of candidate questions are selected. Lastly, the results are optimized via Syntactic Tree Matching (STM) and WordNet-based similarity.

Satapathy et al. (2017) proposed a phonetic-based algorithm to normalize tweets, which is used to enhance the accuracy of polarity detection. It is based on the assumption that humans are able to understand character repetition and substitution in unknown informal words because they automatically shift to the phonetic domain when they read the texts. Therefore, they adopted an ensemble approach that mainly relies on phoneme, and used a lexicon to address other forms of OOV words such as acronyms and emoticons. Given a tweet, all the OOV words are matched with the lexicon to find their In-Vocabulary (IV) forms and corresponding polarity class. Then, any leftover unnormalized tokens are processed by Soundex, which uses homophones to encode texts so that characters with similar pronunciations can be easily matched. Since their objective is to boost polarity detection, the phonetic code of OOV words is matched against the knowledge base SenticNet (Cambria et al. 2022). The final output is fed into a polarity classifier to test the effectiveness of normalization, and yields promising results.

So far, all of the phonetic algorithms used in the above mentioned methods encode words based on their spelling, and thus are only applicable for English or languages written in Latin alphabet. An alternative is to use International Phonetic Alphabet (IPA), which more accurately represents pronunciations and can be used to encode any languages. However, its disadvantage is also evident - such a fine-grained encoding system can be too specific for phonetic matching. Thus, it needs to be simplified accordingly.

Khoury (2015) proposed a phonetic tree-based microtext normalization model. The model determines the probable pronunciation of English words based on their spelling via a radix-tree structured phonetic dictionary. To build the tree, Khoury created a dataset of IPA transcriptions of the English Wiktionary[1]. When encountering an OOV word, the model is able to find the most likely IV word using the uniform-cost search algorithm.

[1] http://www.wiktionary.org/

Jahjah et al. (2016) presented a novel word-searching strategy, built on the idea of sounding out the consonants of a given word. Their algorithm uses spelling and phonetic strategies to extract the base consonant data from misspelled and real phrases. First, the visual signature and phonetic signature of real English words are extracted using a set of rules. The former, similar to spelling-based phonetic algorithms, is defined by the sequence of unrepeated consonants in a word, whereas the latter is represented by the corresponding IPA symbols. The algorithm also extracts simplified reconstructions of the words with vowels re-inserted at the correct places. Subsequently, signatures of a new OOV phrase are generated via a similar set of rules as in the previous step, with special care to the permutations of adjacent consonants and common suffixes. Lastly, the model finds a set of IV words with identical signatures to the OOV word, along with their occurrence probability, and applies several heuristics to find the best matching IV word.

Building upon their previous work (Satapathy et al. 2017), Satapathy et al. (2019b) proposed a cognitively inspired microtext normalizer, named PhonSenticNet, to aid concept-level sentiment analysis for SMS texts. Given a message, a binary classifier is first applied to detect whether microtext is present, aiming to reduce execution time. If true, PhonSenticNet finds the closest match for every concept in the input sentence based on phonetic distance, which is computed using Sorensen similarity algorithm based on both Soundex encoding as well as IPA encoding. Results showed that PhonSenticNet outperforms their previous algorithm.

2.2 Statistical approach

Statistical approaches employ machine learning models such as Hidden Markov Model (HMM) and Support Vector Machine (SVM) to perform text normalization. For this type of normalizers, noisy channel is the most widely-used formulation. Modeling the process of distorting the source into the target, noisy channel is the basis of many NLP applications, e.g., Statistical Machine Translation (SMT), Automatic Speech Recognition (ASR), and spell checking. Intuitively, we can view microtext normalization as a translation task from standard English to informal English. Therefore, the nature of noisy channel makes it suitable for microtext normalization.

Choudhury et al. (2007) constructed a noisy channel model based on HMM for words in standard English to represent all possible variations of their corresponding text language version. Inspired by Rabiner (1989), the core concept of their model is to formulate the transformation from standard English to text language as a noisy channel. Given a sentence s in standard form, the noisy channel decodes it into text language t by

$$\hat{t} = \arg\max{}_s P(s)P(t \mid s)$$

where the language model $P(s)$ encodes which strings in standard form are valid, and the error term $P(t \mid s)$ models how the standard forms are distorted. The former can be easily constructed by exploiting large amounts of unlabeled data, therefore the paper focused on the latter, which is implemented by a letter-to-phoneme conversion HMM. Given a word in standard English, the HMM formulates the graphemes in the word as observation states, and their corresponding phonemes as hidden states. To train and test their method, they created a word-aligned corpus of SMS texts and standard English.

Jiampojamarn et al. (2007) pointed out that the previous letter-to-phoneme conversion systems only support one-to-one letter-phoneme alignment, which leads to difficulties in handling single letters that correspond to two phonemes, or vice versa. This can be partially mitigated by manually constructing a fixed list of merged phonemes prior to the alignment process. The paper, however, proposed a better solution to overcome this limitation - an automatic many-to-many alignment method. Given an input word, a many-to-many aligner establishes the appropriate alignments across the graphemes and phonemes. After this process, the input word is represented as a set of letter chunks, each containing one or two letters aligned with phonemes. The chunk boundaries are determined via a bigram letter chunking prediction model based on instance-based learning (Aha et al. 1991). The prediction model generates all the bigrams in a word, automatically determining whether each of them should be a double letter chunk based on the context. Subsequently, an HMM embedded with a local classifier is applied to find the globally most possible sequence of phonemes of the given word. The local classifier, also based on instance-based learning technique, generates the phoneme candidates for every letter chunk according to the context. Thus, compared to conventional HMM described above, their modified HMM is able to utilize context information from not only the phoneme side but also the grapheme side.

Bartlett et al. (2008) applied a structured SVM model called SVM-HMM model (Altun et al. 2003) to orthographic syllabification, which acts as a sub-system to improve letter-to-phoneme conversion. The paper formulated orthographic syllabification as a sequence labeling problem, employing SVM-HMM to predict tag sequences. SVM-HMM, unlike the standard SVM (Tsochantaridis et al. 2004), employs Markov assumption during scoring, and thus is able to consider complete tag sequences during training. The motivation for using SVM-HMM over HMM is mainly the benefit of the discriminative property in SVM, and leeway to adopt feature representations without any conditional independence assumptions. The paper explored two tagging schemas for syllabification. One is positional tags (NB tags) (Bouma 2003) that label whether each letter occurs at a syllable boundary or not. The other is structural tags (ONC tags) (Daelemans et al. 1997; Skut et al. 2002), which represent the role each letter is playing within the syllable, namely onset, nucleus, and coda. The former is simple, straightforward, and adaptable to different languages. The latter, on the other hand, is able to capture the internal structure of a syllable to improve accuracy, but does not explicitly represent syllable breaks. To mitigate the weakness of the ONC tag schema, the paper combined the two schemas and proposed a hybrid Break ONC tag schema. Experiments show that their model not only outperforms previous syllabification systems, but also improves the accuracy of letter-to-phoneme conversion.

Kaufmann and Kalita (2010) employed a SMT system for Twitter message normalization. Given a tweet, it is first passed through a syntactic normalization module, where sets of rules on the orthographic- and syntactic-level are applied to preliminarily normalize the input sentence. Next, the normalized tweet is fed into the machine translation module, which is implemented by an existing SMT model called Moses (Koehn et al. 2007), to transform the tweet into standard English.

Xue et al. (2011) introduced a method that considers orthographic factors, phonetic factors, contextual factors, and acronym expansions, formulating each of them as a noisy channel. The core concept is that a non-standard term should be similar to its standard form in respect of one or more of these factors, and thus each channel is responsible for one aspect of the distortion that converts the intended form into the observed form. The grapheme channel models the corruption of spellings. The phoneme channel causes distortion in pronunciations. The context channel changes terms around a target term. The acronym channel transforms a phrase into a single term. These channel models are combined using two variations of channel probabilities, namely Generic Channel Probabilities and Term Dependent Channel Probabilities. The former assumes that the probability of a term being emitted through a noisy channel is independent of its standard form, whereas the latter takes into consideration that some standard forms are more likely to be distorted via a certain channel in reality. Experiments show that the two variants achieve similar performance, with the latter being slightly better on SMS dataset.

Han and Baldwin (2011b) theorized that supervised learning is not adept at handling Twitter OOV words due to data sparsity. Hence, they introduced an unsupervised classifier that detects ill-formed words and generates candidate IV words using morphophonemic similarity. Their normalization strategy consists of three stages. First, a confusion set of candidate IV words is generated for each OOV word based on morphophonemic variations, which are produced using lexical edit distance and the double Metaphone algorithm (Philips 2000). Then, based on its confusion set, a linear kernel SVM classifier (Fan et al. 2008) is employed to detect whether the given OOV word is actually an ill-informed word. Lastly, if the OOV word is indeed ill-informed, the most likely candidate is selected based on morphophonemic similarity and context. Although their method outperforms most supervised models, there are limitations. The most prominent one is that it only targets single-token words, and thus is unable to handle phrases and acronyms.

Pennell and Liu (2011, 2014) described a SMT-based system for expanding abbreviations found in informal texts. The method operates in two phases. In the first phase, a character-level SMT model generates possible hypotheses for each abbreviation. By training the model at character level, the model is able to learn the common character abbreviation patterns regardless of their associated words, and thus alleviates the OOV problem. In the second phase, an in-domain language model decodes the hypotheses, choosing the most likely one in the context.

Similarly, Li and Liu (2012) introduced a word-level framework based on a SMT approach, which performs well in translating OOV words into IV words. The framework can be divided into two components. One component is a character-based SMT module that translates non-standard words into standard words by matching their character sequence. The other component is a two-stage SMT module that leverages phonetic information. Non-standard words are first translated into possible phonetic symbols, then mapping to standard words. The candidate word lists generated by the two components are then combined and ranked using a set of heuristic rules.

Liu et al. (2012) described a broad-coverage normalization system that incorporates different aspects of human cognition, including letter conversion, visual priming, and resemblance between string and phone. The system consists of four key components. Given a token, three sub-normalizers respectively produce candidate words based on the cognition aspects, namely, enhanced letter transformation, visual priming, and spell checking. Then, the last component, called candidate combination, uses either word-level or message-level strategy to rank the candidates. The enhanced letter transformation component is a standard noisy channel, where the error term is modeled by a character-level CRF. The enhancement is two-fold. First, features relating to phoneme, morpheme, syllable, and word boundary are used to train the CRF. Second, the training *(word, token)* pairs are chosen according to their global contextual similarity. On the other hand, the visual priming component is also a noisy channel, but uses a novel technique inspired by cognitive behavior. They observed that when humans are familiar with informal texts, they are able to quickly recognize non-standard tokens given only minor visual cues. Following this observation, the priming component computes the error term based on word frequency of the token and the minor visual stimulus, which is LCS between the token and target word divided by the token length. Lastly, for the spell checking component, the open-source Jazzy spell checker[2] is employed. The broad-coverage system outperforms previous state-of-the-art models at both word level and message level.

Zhang et al. (2013) proposed a parser-centric word-to-word normalization method to convert raw informal texts into the correct grammatical version. The main drive for this method is that text normalization performance should be strongly dependent on how useful it is for downstream applications, e.g., parsing. Given an input sequence, a series of replacement generators replace original tokens with normalized ones by producing sets of edit operations. Subsequently, a directed acyclic normalization graph is constructed based on the generated replacements. The graph models two types of dependencies, namely, syntactic consistency of true assignments, and correlations among replacements. The output is determined by using Maximum A Posteriori probability (MAP) inference, which is to select the weighted longest path in the normalization graph. The paper tested the normalizer in two aspect, namely, its effects on dependency parsing and its domain adaptability. Results showed that the proposed normalizer indeed helps improve parsing, and supports domain transfer with low cost.

Wang and Ng (2013) made the observation that aside from standardizing informal words, other normalization operations, such as missing word recovery and punctuation correction, are also crucial for the improvements of downstream NLP tasks. To this end, they presented a punctuation correction method based on dynamic CRF (Sutton et al. 2007), which is described in detail in the text chunking section, and a missing word recovery method based on standard CRF. For the former, the words and punctuation symbols surrounding the focal word are represented as binary features, and fed into a two-layer dynamic CRF model. The first layer assigns the actual punctuation tags, whereas the second layer labels sentence boundaries. For missing word recovery, the paper specifically targeted the omission of "*be*" in English. Hence, a CRF is employed to label every token in the input sentence with a tag that indicates whether the token should be followed by a conjugation of "*be*". Furthermore, they proposed a novel beam-search decoder that can integrate different types of normalization operations, including statistical and rule-based operations. Given an input sentence, the decoder iteratively produces new sentence-level hypotheses, evaluating them to retain the plausible ones, until it finds the best normalization. Each hypothesis is generated by several hypothesis producers. Each of these producers focuses on a different target aspect of informal texts, applying the corresponding type of normalization operation to the sentence. In other words, the objective of beam-search is to find the best pipeline of hypothesis producers. Their method yields significant improvement in both Chinese and English social media texts.

Yang and Eisenstein (2013) suggested that data sparsity of social media texts makes unsupervised learning extremely challenging. To solve this problem, they proposed a unified unsupervised model based on maximum-likelihood framework. Given an informal sentence, a log-linear model is applied to model the string similarity between tokens in tweets and standard English. Since it is an unsupervised, locally-maximized conditional model, typical dynamic programming techniques such as the Viterbi algorithm is not ideal. Instead, they applied the sequential Monte Carlo algorithm (Cappé et al. 2007) for training. The log-linear model is combined with a language model for standard English to output the desired conditional probability.

[2] http://jazzy.sourceforge.net/

Based on the assumption that syllable plays a fundamental role in non-standard word generation in social media, Xu et al. (2015b) viewed syllable as the basic unit and extended the conventional noisy channel approach. Given a non-standard word, it is first segmented into syllables to identify the non-standard syllables. Then, the error term of the noisy channel can be represented as syllable similarity, which is an exponential potential function that combines orthographic similarity and phonetic similarity. The former is measured by edit distance and LCS, whereas the latter is measured by phonetic edit distance and phonetic LCS (PLCS) based on letter-to-phone transcripts. Results showed that the syllable-based approach indeed yields significant improvement. Additionally, this method has the advantage of being robust and domain independent.

2.3 Neural network approach

In recent years, deep learning models are widely used in the filed of NLP (Otter et al. 2020; Mao et al. 2019), as they are able to automatically extract features from input text. Neural network approaches make use of this advancement in microtext normalization domain. As stated in the previous section, microtext normalization can be viewed as a translation problem. Hence it is a natural progression to employ the encoder-decoder framework commonly adopted by NMT (Yang et al. 2020). Additionally, since the attention of normalizing microtext has gradually shifted from word level to character level, neural networks are frequently utilized to extract character-level features.

Chrupała (2014) introduced a semi-supervised text normalization model. They pointed out a weakness of the widely used noise channel decomposition in microtext normalization - texts in normalized form of source domain are limited, and standard text resources vary across domains. Thus the target language model is in fact not as easy to estimate as one imagines. Therefore, their model serves as an alternative that utilizes a large amount of unlabeled data from a source domain. To achieve this, normalization is formulated as a string transduction problem (Chrupała 2006), which is discussed in detail in the lemmatization section. Using labeled data, a CRF learns the sequence of edit operations that transforms the tweets to standard English. Additionally, a Simple Recurrent Network (Elman 1991) is employed to induce character-level embeddings from unlabeled Twitter data, which are used as features in the CRF transduction model. Their model is able to yield accurate results with less training data and without lexical resources.

Leeman-Munk et al. (2015) proposed a deep learning model to solve the W-NUT (see Table 1 for W-NUT) Lexical Normalization for English Tweets challenge. Their model consists of three components, a normalizer, a flagger that identifies words to be normalized, and a conformer that smooths out the simple errors from the normalizer. The normalizer is implemented as a character-level three-layer feedforward neural network, which takes in the input word and predicts a possible standard version. The flagger has the same architecture as the normalizer and works in parallel with the normalizer. The input word is replaced by the generated word only if the flagger tags it as to be normalized. Since the normalizer functions at character level, it is possible that the predicted word is one character off. To address this error, a conformer is constructed using a dictionary collected from the gold standard training data. If the word generated by the normalizer is not in the dictionary, the conformer replaces it with the most similar word in the dictionary based on Levenshtein edit distance.

Taking inspiration from the advances in neural Machine Translation (MT), Lusetti et al. (2018) proposed a character-level encoder-decoder model to normalize multilingual

SMS messages. The model employs a Bidirectional Long Short-Term Memory (Bi-LSTM) (Graves et al. 2013) as an encoder, an LSTM (Hochreiter and Schmidhuber 1997) as a decoder, and a soft attention mechanism (Bahdanau et al. 2014). They further introduced two methods to augment the model. First is to fuse it with a word-level language model of the target language using the synchronization mechanism (Ruzsics and Samardzic 2017). Second is to enhance the training set with additional target-side data.

Similarly, Satapathy et al. (2019a) tested different variants of the Sequence-to-Sequence (Seq2Seq) normalization framework, and to what extent they can improve the performance of sentiment analysis on informal texts. Specifically, they experimented with four types of character-level encoder-decoder architectures, namely LSTM, LSTM with attention, Gated Recurrent Units (GRU) (Cho et al. 2014), and LSTM with Convolutional Neural Network (CNN) as feature extractor (Kim 2014). Experiments show that all the normalizers above help boost the accuracy of sentiment analysis on social media and SMS texts. The ranking of effectiveness from high to low is GRU, LSTM with attention, LSTM with CNN, and LSTM.

Lourentzou et al. (2019) proposed a hybrid word-character attention-based encoder-decoder model for text normalization. Given a sentence, the tokens are first passed through a word-based Seq2Seq model to capture semantic meaning and long-term contextual dependencies. Any detected OOV tokens are marked and passed on to a secondary character-based Seq2Seq model, which outputs the possible IV tokens and corresponding confidence. The character-based Seq2Seq model is trained using synthetic adversarial data that aims to capture errors commonly found in social media texts.

2.4 Summary

Microtext normalization is in fact a complex task containing many sub-problems, e.g., misspelled words, acronyms, phonetics-based spelling alterations, real-word spelling errors, missing words and punctuations, etc.

In summary, among linguistic approaches, the system proposed by Pennell and Liu (2010) is advantageous in that it does not require external lexicons, however, it relies on hand-crafted rules to generate candidates. Jose and Raj (2014) incorporated a parser as subsystem, effectively capturing lexico-syntactic information. However, their method heavily relies on manually complied database. The representation of string similarity is a focal point of linguistic approaches. Desai and Narvekar (2015) used Levenshtein edit distance to measure similarity, which cannot model the human intuition behind microtext. Mittal et al. (2014), on the other hand, applied multiple spelling-based phonetic matching algorithms to compute similarity. However, their method is language-specific. IPA phonetic matching can be applied to any language, while it suffers from over-specificity (Khoury 2015). Jahjah et al. (2016) mitigated this weakness by incorporating spelling features combined with IPA phonetic matching. Brody and Diakopoulos (2011) and Satapathy et al. (2017, 2019a) all targeted sentiment analysis. Brody and Diakopoulos (2011) relied a set of simple rules for elongated words, and thus limited in use. The latter two models are able to handle various types of OOV words, among which Satapathy et al. (2019b) has the advantage of incorporating IPA encoding.

For statistical approach, most models formulate microtext normalization as a noisy channel problem, as first implemented by Choudhury et al. (2007). A limitation of their method is that it can only map one letter to one phoneme. Jiampojamarn et al. (2007) implemented a many-to-many letter-phoneme alignment that can mitigate this problem, whereas their solution relies on hand-crafted conversion rules and only works on fixed window size of letters. Bartlett et al. (2008) and Xu et al. (2015b) addressed this limitation by utilizing syllable. The former uses syllable tagging as a sub-system, whereas the latter considers syllable as the basic unit for noisy channel. Han and Baldwin (2011b) and Yang and Eisenstein (2013) both utilized unsupervised learning to mitigate the data sparsity problem for social media text. However, their methods cannot handle phrases and acronyms. The parsing-centric method proposed by Zhang et al. (2013) is advantageous in aiding downstream task, however, it has the same weakness of only dealing with single-token word-to-word conversion. This limitation can be addressed by SMT-based approach. Kaufmann and Kalita (2010) directly implemented an existing SMT system for microtext normalization. Following this approach, Pennell and Liu (2011) presented a character-level SMT system that can better handle abbreviation. Li and Liu (2012) further improved such method by combining character-level and phonetic SMT systems. On the other hand, Xue et al. (2011) tackled the difficulty of abbreviation by using multiple noisy channels, each for one aspect of microtext. Similarly, Liu et al. (2012) also incorporated multiple noisy channels, and took it one step further by designing each noisy channel specifically for its function. Lastly, Wang and Ng (2013) confronted the niche aspects of microtext normalization, i.e., missing word recovery and punctuation correction.

With neural network approaches, Chrupała (2014) addressed the target domain problem of MT-based approach by proposing a semi-supervised alternative. However, such method only targets single-token word. Leeman-Munk et al. (2015) circumvented both of these limitations by using a character-level pipeline model, which conversely can introduce propagated errors. Lusetti et al. (2018), Satapathy et al. (2019a) and Lourentzou et al. (2019) all made use of the encoder-decoder structure with various neural networks, which can be considered as an MT-based approach. Nonetheless, the target domain problem is less prominent nowadays, because a) more annotated datasets are available, and b) deep learning approach is more adept at extracting generalized information than traditional SMT systems.

Table 2 presents a summary of the introduced methods and their concerns. Notably, most linguistic and statistical methods use a combination of orthographic matching and phonetic matching by computing their corresponding similarities. This is expected, as each provides complementary information that the other cannot address. There are methods that use neither, formulating their target problems as sequence prediction tasks instead, using either graphical models or neural networks. For some approaches, contexts are taken into consideration by incorporating syntactic features (Fossati and Di Eugenio 2008; Kaufmann and Kalita 2010; Zhang et al. 2013), or using attention mechanism (Lusetti et al. 2018; Satapathy et al. 2019a; Lourentzou et al. 2019). Additionally, syllable-level and character-level models are able to learn more fine-grained patterns than word-level ones, and thus are gaining popularity in recent years, especially for the latter. Since microtext normalization is a field with abundant raw text, we also mark out methods that are able to utilize unlabeled data. Finally, there are specific sub-problems that were addressed by a part of the reviewed linguistic and statistical methods, namely, abbreviation and acronym and sentiment-related error. Table 3 shows the performance of the introduced methods, organized by their target tasks. On the SMS-C dataset, Li and Liu (2012) obtained the highest top-1 accuracy, whereas Liu et al. (2012) obtained the highest top-20 accuracy. Zhang et al. (2013) achieved the best performance on LexNorm1.1, and Xu et al. (2015b) on LexNorm1.2.

To sum up, linguistic approaches build upon simple text-level and phonetic-level matching algorithms, utilizing hand-crafted rules, lexicons or knowledge bases. As mentioned above, it is adequate at targeting specific problems, e.g., IPA-based phonetic

Table 2 The comparison between different Microtext normalization methods

Method		OM	PM	Context	Syllable	Character	UD	Abb.	Sentiment
Linguistic	Pennell and Liu (2010)							✓	
	Brody and Diakopoulos (2011)	✓							✓
	Jose and Raj (2014)	✓	✓					✓	
	Mittal et al. (2014)	✓	✓						
	Desai and Narvekar (2015)	✓	✓						
	Satapathy et al. (2017)	✓	✓					✓	✓
	Khoury (2015)		✓						
	Jahjah et al. (2016)		✓						
	Satapathy et al. (2019b)		✓						✓
Statistical	Choudhury et al. (2007)	✓	✓				✓		
	Jiampojamarn et al. (2007)	✓	✓						
	Bartlett et al. (2008)	✓			✓				
	Kaufmann and Kalita (2010)	✓		✓					
	Xue et al. (2011)	✓	✓	✓			✓	✓	
	Han and Baldwin (2011b)	✓	✓	✓			✓		
	Pennell and Liu (2011)			✓		✓	✓	✓	
	Li and Liu (2012)	✓	✓			✓	✓	✓	
	Liu et al. (2012)	✓	✓			✓	✓	✓	
	Zhang et al. (2013)	✓		✓					
	Wang and Ng (2013)			✓					
	Yang and Eisenstein (2013)	✓					✓		
	Xu et al. (2015b)	✓	✓		✓		✓		
Neural	Chrupała (2014)					✓	✓		
	Leeman-Munk et al. (2015)					✓		✓	
	Lusetti et al. (2018)			✓		✓		✓	
	Satapathy et al. (2019a)			✓		✓		✓	
	Lourentzou et al. (2019)			✓		✓		✓	

OM stands for orthographic matching based on string similarity. PM stands for phonetic matching based on phonetic similarity. UD stands for unlabeled data. Abb. stands for abbreviation

matching, sentiment, and context provided by metadata. However, its drawbacks are prominent. Rule-based and lexicon-based methods require a lot of human effort, hence unable to adapt to other language domains. Statistical approach, on the other hand, is much less labor-intensive. This type of models regard microtext normalization as a SMT problem or a sequence labeling problem. Although most of them still rely on feature engineering, they pave the way for neural network approaches. With the help of neural networks, context-sensitive character-level features can be easily extracted by Seq2Seq models. It is also adaptable to other languages.

Table 3 The performance of the introduced microtext normalization methods

Task	Method	Experiment 1			Experiment 2		
		D	R	M	D	R	M
Nonstandard	Pennell and Liu (2010)	Twitter API + SMS-G	58.46	top-1 acc.	Twitter API + SMS-G	75.38	top-3 acc.
	Jose and Raj (2014)	SMS-G	95.41	acc.			
	Khoury (2015)	SMS-L	30.2	top-1 acc.	SMS-L	59.7	top-5 acc.
	Jahjah et al. (2016)	SMS-L + SMS-C	56.6	top-1 acc.	SMS-L + SMS-C	73.6	top-5 acc.
	Satapathy et al. (2019b)	NUS SMS	92.75	acc.	Twitter API	72.88	acc.
	Kaufmann and Kalita (2010)	Twitter API	79.85	BLEU			
	Pennell and Liu (2011)	Twitter API	75.57	top-1 acc.	Twitter API	92.53	top-20 acc.
	Choudhury et al. (2007)	SMS-C	58	top-1 acc.	SMS-C	86	top-20 acc.
	Xue et al. (2011)	SMS-C	96	acc.	Twitter API	96	acc.
	Han and Baldwin (2011b)	SMS-C	75.5	F1	LexNorm1.1	75.3	F1
	Li and Liu (2012)	SMS-C	71.96	top-1 acc.	SMS-C	83.11	top-20 acc.
	Liu et al. (2012)	SMS-C	64.36	top-1 acc.	SMS-C	91.75	top-20 acc.
	Liu et al. (2012)	LexNorm1.1	69.81	acc.			
	Yang and Eisenstein (2013)	LexNorm1.1	82.09	F1	LexNorm1.2	82.06	F1
	Yang and Eisenstein (2013)	SMS-L	73	F1			
	Xu et al. (2015b)	LexNorm1.1	85.3	F1	LexNorm1.2	86.08	F1
	Chrupała (2014)	LexNorm1.1	4.5	WER			
	Leeman-Munk et al. (2015)	W-NUT	81.49	F1			
	Lourentzou et al. (2019)	W-NUT	83.94	F1			
	Lusetti et al. (2018)	WUS + Sw-SMS	87.61	acc.			
Polarity	Brody and Diakopoulos (2011)	Twitter API	93	acc.			
	Satapathy et al. (2017)	Twitter API	81.59	acc.			
	Satapathy et al. (2019a)	NUS SMS	83.5	acc.			
L2P	Jiampojamarn et al. (2007)	CMUDict	65.6	acc.	CELEX Eng	83.6	acc.
	Bartlett et al. (2008)	CELEX	89.99	acc.			

Table 3 (continued)

Task	Method	Experiment 1			Experiment 2		
		D	R	M	D	R	M
QA system	Mittal et al. (2014)	FAQ corpus[a]	75	acc.			
Missing tokens	Wang and Ng (2013)	NUS SMS	66.54	BLEU			
Parsing	Zhang et al. (2013)	SMS-C	81	F1	LexNorm1.1	91.9	F1

D denotes dataset. R denotes result. M denotes measure. Nonstandard denotes Normalizing informal OOV words. L2P denotes letter-to-phoneme. Twitter API stands for data randomly collected from Twitter. SMS-G stands for SMS messages generated by volunteers. WER is word error rate

[a]http://irsi.res.in/fire/faq-retrieval/data.html

Table 4 Widely used corpora for SBD

	Dataset	Source	Reference
Text	BC	The Brown Corpus	Francis and Kucera (1979)
	WSJ	Wall Street Journal	Marcus et al. (1993)
	SMC	Twitter text corpus	Rudrapal et al. (2015)
	MG	Texts in Modern Greek	Stamatatos et al. (1999)
	GRS	Gold Standard Rules	Sadvilkar and Neumann (2020)
	MIMIC	Medical Information Mart for Intensive Care corpus	Johnson et al. (2016)
	FV	Fairview Health Services database	Knoll et al. (2019)
Prosody	T-News	Speech transcripts from news program	Stevenson and Gaizauskas (2000)
	T-CTL	Prepared speech transcripts	Treviso et al. (2017)
	T-MCI	Impaired speech transcripts	Treviso et al. (2017)
	T-R	Transcripts of BBC and radio news	Gotoh and Renals (2000)
	BN	Broadcast news transcripts	Liu et al. (2004)
	CTS	Conversational telephone speech transcripts	Liu et al. (2004)

However, the neural network approaches still have much room for improvement. To our knowledge, there is not yet a deep learning normalizer that directly addresses the data sparsity problem caused by phonetics-based alterations. It can potentially lead to performance drop for languages that are rich in such alterations on social media, such as Chinese. Thus, incorporating phonetic features or a phonetic-level encoder can be a potential direction for future work.

Furthermore, the use of microtext is strongly affected by the geolocation and dialect of users. For instance, when expressing laughter in Latin alphabets, there is a tendency to use *"lol"* for English speakers, *"hhh"* for Chinese speakers, *"www"* for Japanese speakers, *"kkk"* for Korean speakers, *"555"* for Thai speakers, etc. Metadata that provides such information is scarcely utilized in existing methods. As such, this would also be an interesting direction to explore.

3 Sentence boundary disambiguation

SBD, which decides where sentences begin and end in raw texts, is an important yet overlooked pre-processing task for many NLP applications. Downstream tasks such as machine translation (Matsoukas et al. 2007; Zhou et al. 2017) and document summarization (Jing et al. 2003; Boudin et al. 2011) rely on predetermined sentence boundaries for good performance. Sentence segmentation is seemingly easy through identifying punctuation marks. However, there are some notable ambiguities that often occur in text. For instance, aside from indicating the end of a sentence, a period can also appear in abbreviations or as decimal point. In a more complex scenario where an abbreviation is the last token of a sentence, the period simultaneously serves as a part of the abbreviation as well as the full-stop of the sentence.

There are two widely-used corpora for SBD, the Brown Corpus (Francis and Kucera 1979) and the WSJ corpus from the Penn Treebank project (Marcus et al. 1993). In the

Brown Corpus, 90% of periods occur at the end of sentences, and 10% at the end of abbreviations, among which about 1% to 2% are in both, whereas in the WSJ corpus, about 47% of periods appear in abbreviations. Thus, the ambiguity level largely depends on corpora and the associated domains. Widely applied datasets are listed in Table 4. The evaluation metrics commonly used are accuracy (or error rate) and F-measure.

In this paper, we divide the existing SBD algorithms into three categories: word-based approaches, syntax-based approaches, and prosody-based approaches.

3.1 Word-based approach

The word-based approaches largely rely on morpholexical information, e.g., suffix, spelling, capitalization, and word length to segment sentences. For early works, rule-based approaches, *n*-gram probabilistic methods, and lexical lookup methods are commonly employed to predict the type of the words surrounding a period, thus determining whether the period marked the end of the sentence.

Grefenstette and Tapanainen (1994) were the first to propose a rule-based algorithm to resolve the ambiguities caused by the usage of periods. Specifically, a set of rules are manually created using regular expressions to represent possible patterns where periods that do not occur as full-stops. The system then matches the surrounding context of every period in texts against the regular expressions to predict whether it is a sentence boundary. Their system obtains reasonably high accuracy with great computational efficiency.

To enable automated extraction of rules, Stamatatos et al. (1999) applied transformation-based learning to SBD. Compared to the original transformation-based algorithm (Brill 1995), which is discussed in detail in the POS tagging section, their method limits the number of possible transformations. This is achieved by maintaining two sets of rules for each punctuation mark. Initially, all punctuation marks are considered to be full-stops. Then for each of them, the system automatically learns a set for rules that triggers the removal of a sentence boundary, and a set of rules that triggers the insertion of a sentence boundary. This mechanism ensures that the maximum number of transitions for each punctuation mark is two.

Sadvilkar and Neumann (2020) developed a rule-based SBD system called PySBD, where the reasoning mechanism of their system are explainable. The performance is comparable to statistical models. Unlike the other supervised SBD methods, PySBD is trained and evaluated with the Golden Rule Set (GRS). GRS is a language specific corpus designed for SBD. It contains sets of hand-crafted rules over a variety of domains that are carried out in a pipeline fashion.

Despite the decent accuracy, there are some drawbacks to rule-based methods. Firstly, periods exhibit absorption properties, meaning when multiple periods occurs they are often marked as one. Therefore, it is challenging to build a comprehensive set of rules where they do not contradict each other. Furthermore, such systems are often developed using a specific corpus, thus further application to a corpus in another language or domain is difficult. Therefore, many machine learning methods are proposed for the SBD tasks to address these shortcomings.

To reduce the labor efforts in developing hand-crafted rules and features, Reynar and Ratnaparkhi (1997) proposed a SBD system based on MaxEnt (Ratnaparkhi 1996). It requires only simple information about the candidate punctuation marks. For each candidate punctuation mark, the system utilizes morpholexical features of its trigram tokens, estimating their joint probability distribution as a sentence boundary. The performance of their system is comparable to those which require vastly more resources.

Schmid (2000) proposed an unsupervised learning method by manually listing all the possible scenarios where a period may denote a decimal point or an abbreviation. For each of the listed scenarios, a probability model is created accordingly to predict whether the period is a full-stop. Unlike previous token-based methods, which rely on the focal token itself and its local context to determine whether it is an abbreviation, their model computes the distribution of possible abbreviations by scanning through the whole corpus. This method can be referred to as a type-based approach (Kiss and Strunk 2002).

Similarly, Kiss and Strunk (2002) introduced an innovative unsupervised type-based system that requires less human effort. Their approach views the identification of abbreviations as a collocation detection problem, which can be solved using log likelihood ratio. Specifically, an abbreviation can be seen as a collocation of the abbreviated word and the following period.

Kiss and Strunk (2006) further expanded this system to create the Punkt system, which has a more complex structure. First, the input text is processed by the previous type-based algorithm to obtain the initial annotation, which marks a period as either part of an abbreviation, part of an ellipsis, or sentence boundary. Subsequently, a token-based classifier is applied to revise the initial annotation as well as to determine whether a period is both part of an abbreviation or ellipsis and the end of sentence, thus producing the final annotation. In the token-based classifier, log likelihood ratio is used for two heuristic. One is the collocation heuristic, which takes a pair of word surrounding a period and tests whether a collocation tie exists between them. The other is the frequent sentence starter heuristic, which searches for word types that form a collocation with a preceding sentence boundary based on the results from the type-based classifier. The frequent sentence starter heuristic helps counterbalance the collocation heuristic, which in some scenarios falsely identifies collocation across a sentence boundary.

Gillick (2009) proposed a SBD system based on SVM. The system takes the trigram contexts of periods as input. Specifically, for a trigram of the form "*L.R*", the problem is defined as the conditional probability of the binary sentence boundary class *s*, $P(s \mid$"*L.R*"). The system utilizes morpholexical features, e.g., capitalization and word length to train the SVM. To reduce the error rate caused by words such as *U.S* and *N.Y*, the system adopts the type-based approach, taking into account the log count of lowercased appearances of *L* and *R* in the input texts. Aside from the proposed system, the paper also makes the observation that the majority of errors in SBD occur when an abbreviation is the end of sentence.

Rudrapal et al. (2015) focused on the problem of SBD in social media texts and explored three machine learning algorithms for the task, namely CRF, Naïve Bayes (NB), and Sequential Minimal Optimization (SMO) (Platt 1999). Since social media texts likely use informal languages, identifying sentence boundaries in such texts is a even greater challenge. To mitigate the ambiguities, the corpus is first tokenized, using the tokenizer in CMU Twitter POS tagger (O'Connor et al. 2010; Gimpel et al. 2010). Subsequently, the machine learning algorithms are trained on the basic textual features of the tokenized corpus. The CRF method is able to extract simultaneously correlated lexical features, thus easier to incorporate different knowledge sources. The SMO method, on the other hand, is an iterative optimization approach to train SVM. Results show that SMO yields the best performance on social media texts, whereas NB comes close in the second place. On the Brown corpus, both NB and SMO obtain good performance, but the former is more accurate. CRF, however, performs marginally worse than the other two algorithms.

Riley (1989) first introduced a tree-based algorithm to SBD, where a regression tree is generated using a set of hand-crafted morpholexical features from the trigram context of words with periods in them to identify decimal points and abbreviations. Although their model achieves accurate results, the model requires a vast amount of training data, making it too costly for a pre-processing task and unrealistic for languages and domains with limited resources.

Wong and Chao (2010); Wong et al. (2014) applied an incremental algorithm on SBD. Their system is based on i^{+}Learning principle, which is an incremental decision tree learning algorithm, making it flexible to changes and suited for online learning. The algorithm can be divided into two phases. First, the system constructs a top-down binary decision tree offline using the initial training data. The resulting tree acts as an optimal base for the second phase, which is a online procedure that adopts the tree transposition mechanism of Incremental Tree Induction (ITI) (Utgoff et al. 1997) as a bias to grow and revise the base tree. This method dynamically revises the tree according to the new incoming data while preserving the essential statistical information. Thus it is able to adapt to texts in a different language or domain without the need to retrain from scratch. Their system is trained on morpholexical features extracted from trigram contexts.

All of the aforementioned SBD systems, with the exception of Kiss and Strunk (2002, 2006), use *n*-gram based technique to extract textual information, which leads to sparse vector space problems. To address this, Treviso et al. (2017) suggested word embedding as an alternative and verified which embedding induction method works best for SBD. They investigated Word2Vec (Mikolov et al. 2013), Wang2Vec (Ling et al. 2015a), and FastText (Bojanowski et al. 2016). They used Continuous Bag-Of-Words (CBOW) and Skip-gram to train the vectors, respectively. They then tested the embeddings on a Recurrent Convolutional Neural Network (RCNN) (Treviso et al. 2016) for sentence segmentation. Experiments show that Word2Vec consistently performs better than the other two methods. Additionally, the Skip-gram strategy generally yields better results than the CBOW for the Wang2Vec and the FastText, whereas for Word2Vec the better strategy depends on the corpus. Furthermore, they also compared the model performance between using only the extracted morpholexical features and adding morphosyntactic features for SBD. Interestingly, the results show that the explicitly added features do not make a difference in terms of accuracy. They theorized that the word embedding alone carries sufficient morphosyntactic information for SBD.

Knoll et al. (2019) utilized both word embeddings and character embeddings to further capture morpholexical features. Based on the observation that previous SBD systems perform poorly on a domain specific corpus such as clinical texts, they proposed a deep learning algorithm to address this problem. First, the input text is tokenized and transformed into word embeddings using FastText. The text is also fed into a Convolutional Neural Network (CNN) layer (Collobert et al. 2011) to obtain character embeddings, which is summed with the word embeddings. The final word representation is passed through a Bidirectional Long Short-Term Memory (Bi-LSTM) layer (Graves et al. 2013) and a sigmoid-activated dense layer to output the log-probability of a word being the start of a sentence. They tested their algorithm on the Medical Information Mart for Intensive Care (MIMIC) corpus (Johnson et al. 2016) and a dataset drawn from the Fairview Health Services (FV). Results suggest that the deep learning approach indeed outperforms the traditional ones, especially on corpora from different domains and corpora where sentences are often not terminated by punctuation marks.

3.2 Syntax-based approach

It is difficult for word-based approaches to robustly capture contexts larger than the focal token. Syntax-based SBD systems can solve this problem by utilizing POS tags. Intuitively, we know that the number of possible POS patterns for a bigram is much less than the number of possible word patterns, thus lessening sparsity problems. Additionally, the syntactic function of a word makes a big difference when identifying abbreviations. For instance, when predicting whether a capitalized word following a period is a proper name or a common word, taking into account the POS tags of its trigram context is more effective than relying only on their morpholexical properties.

Palmer and Hearst (1994) were the first to incorporate POS tagging in the task of SDB. They proposed an efficient and portable system based on feed-forward neural network. The core idea is to use POS probabilities of the tokens surrounding a punctuation mark as input to the feed-forward network, which outputs an activation value to determine what label to assign to the punctuation mark. First, the system uses a slightly modified version of the PARTS POS tagger (Church 1988), which also produces the frequency counts of the POS tags associated with each token. The system then maps them into a more generalized set of 18 POS categories. Each input token in the n-gram context around a punctuation is then represented by a descriptor array indicating their probability distributions for these 18 categories, in addition of two flags that mark whether the word is capitalized and if it follows a punctuation mark. Subsequently, these descriptor arrays are fed into a fully-connected hidden layer with a sigmoid activation function, and then into an output unit to decide whether the punctuation marks are full-stops. The system also introduces two adjustable thresholds to leave room for difficult ambiguities. When the output score falls under the first threshold, the punctuation mark is not a sentence boundary. When the score is higher than the second threshold, it is a sentence boundary. If the score is in between these two thresholds, that means the system is under-informed to make a confident prediction and it'll be marked accordingly for later use.

Based on this algorithm, Palmer and Hearst (1997) further developed a SBD system called Satz, where the aforementioned n-gram descriptor arrays containing POS information can be fed into either a neural network as in their previous work, or a decision tree. The learning algorithm chosen for the system is a c4.5 (Salzberg 1994) decision tree induction program, which iteratively constructs the tree using the descriptor arrays as input attributes. Each leaf node of the decision tree represents the value of the goal attribute, which in this case indicates whether a punctuation mark is a sentence boundary. After the decision tree is built, the algorithm prunes it by recursively examining each sub-tree to reduce errors and overfitting. Experiments show that the tree-based learning method achieves comparable accuracy as the feed-forward neural network. However, there are problems with the n-grams of generalized POS categories. First, the generalized categories are far sparser than the traditional Penn Treebank POS tags, thus requiring more training data. Furthermore, since words outside of the n-gram have no influence on the prediction, the n-gram must be of sufficient length to capture syntactic information. In the Satz System, n is set to be 6, which is also much sparser than the commonly adopted bigrams and trigrams. The reason behind these weakness is that, this method is built on the premise that sentence boundaries must be obtained before POS tagging. Thus, to utilize syntactic information, the original POS tags have to replaced by the generalized POS categories.

This dilemma is solved by Mikheev (2000), who suggested that POS taggers do not necessarily require predetermined sentence boundaries to operate. The simple solution is to break the input text into short text-spans so that it is easier for POS taggers to handle. Based on this notion, he proposed a SBD system using POS tagging framework. To achieve this, he made some minor adjustments to the Brown Corpus and the WSJ corpus. First, the period in abbreviation is tokenized separately from the rest of the abbreviations. Second, all the periods are marked accordingly to three types of tags, namely full-stop, abbreviation, or both. With this setting, SBD can be performed using a POS tagger, which is able to fully make use of the local syntactic information. The POS tagger chosen for this system is based on HMM and MaxEnt, which will be discussed in detail in the POS tagging section. He further introduced a document-centered approach (Mikheev 2002), which can also be regarded as type-based. When using this approach, the system scans the entire document for contexts where the focal token is marked unambiguously. This approach is proven to be effective in distinguishing whether a capitalized word is a proper name or a common word, which works in complement with the POS tagger to determine sentence boundaries.

Stevenson and Gaizauskas (2000) applied memory-based learning to identify sentence boundaries in transcripts produced by an ASR system. It is more challenging than SBD with standard texts. For instance, the text generated by an ASR system is generally unpunctuated, in single case, and likely to contain transcription errors. Their algorithm is based on the Timbl memory-based learning algorithm (Daelemans et al. 2003), which memorizes a set of training examples. It classifies new instances by assigning them the class of the most similar learned instances. The system adopts both POS tags and morpholexical features such as capitalization and stop word flag. Results show that the proposed algorithm cannot effectively address the difficulties in ASR transcripts.

Following the work of Reynar and Ratnaparkhi (1997), Agarwal et al. (2005) proposed a MaxEnt classifier that incorporates both lexical and syntactic information. Specifically, they assigned each token with a binary End-of-Sentence tag and a POS tag, formulating SBD as a sequential labeling problem. Similar to the previous MaxEnt model, their classifier is also trained on the features of trigram contexts. Their work also concluded with the best tested feature set for the MaxEnt classifier.

3.3 Prosody-based approach

Prosody-based approaches are proposed specifically for SBD in ASR. Previous methods (Stevenson and Gaizauskas 2000; Treviso et al. 2017) toke a word-based approach, using only textual features from speech transcripts. However, transcripts tend to be erroneous, and thus are unable to provide reliable cues to segment sentences. Prosody-based approaches, on the other hand, incorporate prosodic information, e.g., pitch, pause length, pre-pausal lengthening, and energy patterns. Such prosodic features are assumed to be able to compensate the lack of punctuation and capitalization in speech. Furthermore, sentence structures are often more elusive in spoken language. A sentence-like unit (SU) in speech can be a sentence, part of a sentence, or a semantically complete component in a sentence (Strassel 2003). Therefore, SBD in ASR faces the additional challenge of determining whether an SU is an actual sentence.

Gotoh and Renals (2000) presented two finite state models to statistically extract sentence boundary information from program scripts, as well as transcriptions of broadcast speech produced by a vocabulary speech recognition system. The former is an *n*-gram type language model trained by the program scripts. The latter is a pause duration model, which is a statistical prosody model estimated from the outputs of an ASR system aligned with their corresponding program transcripts. The joint probability of a sequence of prosodic features are combined the corresponding sequences of word- and class-tokens to enhance the accuracy.

Liu et al. (2004) first built a HMM-based SBD system for continuous speech. Then they proposed a MaxEnt posterior probability model, which aims to address two main shortcomings of the HMM system. Firstly, the training objective of HMM is to maximize the joint probability of observed and hidden events, which is not the best criterion to reduce classification error. Secondly, the *n*-gram language model underlying the HMM transition model requires a significant increase in the number of model parameters to incorporate highly correlated features. The former is solved by replacing the generative model with a posterior probability model. The latter is addressed by MaxEnt framework, which allows for a large number of overlapping features. Aside from word *n*-gram and its syntactic information, they also utilized prosodic features around each word boundary, e.g., duration, pitch, and energy patterns. The system is evaluated using the NIST[3] SU error rate. Results suggest that the proposed model and HMM have complementary strengths and weaknesses. The MaxEnt model is able to achieve better accuracy, whereas the HMM makes more effective use of prosodic features, and degrades less with word recognition errors.

Taking it one step further, Liu et al. (2005) applied CRF to SBD in speech, comparing its performance with the HMM model and the MaxEnt model. CRF has the advantages of being discriminatively trained and able to model the entire sequence. As in their previous work, textual and prosodic information is used to train the models. Results show that CRF indeed performs marginally better than the HMM and MaxEnt. The only shortcoming of CRF model is that, compared to other two models, it takes longer to train as the number of features increases.

Liu et al. (2006) found that a SU in speech is less frequently a sentence than a non-sentence unit. Thus, a prosody model must be able to deal with the imbalanced dataset distribution. To address this problem, they modeled prosodic information with a decision tree classifier in their HMM-based system (Liu et al. 2004). They also investigated different types of sampling approaches, concluding that random downsampling is the most advantageous.

3.4 Summary

In conclusion, existing SBD systems can be categorized into word-based, syntax-based, and prosody-based approaches. Table 5 presents a summary of the introduced systems, and the features and methods they used. In this paper, for convenience, if a SBD system only use the word itself as lexical features, it is not counted as a method using morpholexical features. A comparison of all the system performance can be found in Table 6. The datasets used are listed in Table 4. On the Brown Corpus, word-based method proposed by Wong and Chao (2010) achieved the highest accuracy, exceeding the best syntax-based method (Mikheev 2002) by 0.26%. On the WSJ dataset, the best word-based method (Gillick 2009) obtained 99.75% accuracy, whereas the best syntax-based method (Mikheev 2002) obtained 99.55%. For prosody-based approaches, Liu et al. (2005) obtained the lowest NIST SU error rate.

[3] http://www.nist.gov/speech/tests/rt/rt2003/fall/

Table 5 The comparison between different SBD methods

Method		Morpholexical	Syntactic	Prosodic	TK	TP	Neural	ASR
Word	Grefenstette and Tapanainen (1994)	✓			✓			
	Stamatatos et al. (1999)	✓			✓			
	Sadvilkar and Neumann (2020)	✓			✓			
	Reynar and Ratnaparkhi (1997)	✓			✓			
	Schmid (2000)	✓				✓		
	Kiss and Strunk (2002)	✓				✓		
	Kiss and Strunk (2006)	✓				✓		
	Gillick (2009)	✓				✓		
	Rudrapal et al. (2015)	✓			✓			
	Riley (1989)	✓			✓			
	Wong and Chao (2010)	✓			✓			
	Wong et al. (2014)	✓			✓			
	Treviso et al. (2017)	✓					✓	✓
	Knoll et al. (2019)	✓					✓	
Syntax	Palmer and Hearst (1994)	✓	✓		✓			
	Palmer and Hearst (1997)	✓	✓		✓			
	Mikheev (2000)		✓		✓			
	Mikheev (2002)	✓		✓		✓		
	Stevenson and Gaizauskas (2000)	✓	✓			✓		✓
	Agarwal et al. (2005)	✓	✓		✓			
Prosody	Gotoh and Renals (2000)	✓		✓	✓			✓
	Liu et al. (2004)		✓	✓	✓			✓
	Liu et al. (2005)		✓	✓	✓			✓
	Liu et al. (2006)		✓	✓	✓			✓

TK denotes token-based method. TP denotes type-based method. ASR denotes automatic speech recognition

Word-based systems utilize morpholexical features, e.g., capitalization, suffix, word length, etc, which can be further divided into token-based, type-based, and neural network based. Given a focal word, token-based methods use rules or machine learning models to determine sentence boundary based on the features of the local context. Among rule-based systems, the work of Stamatatos et al. (1999) could automatically generate rules, and Sadvilkar and Neumann (2020) applied sets of accurate rules in pipeline. Since the rule-based methods are rigid and labor intensive, Reynar and Ratnaparkhi (1997) and Rudrapal et al. (2015) alleviated this by using statistical models that require simple features. Riley (1989) proposed the first tree-based SBD system, which is more comprehensible to human. To reduce the computation cost, Wong and Chao (2010); Wong et al. (2014) applied an incremental tree-based algorithm, which is also more flexible and suited for online learning. Type-based methods, compared to token-based ones, not only consider the morpholexical features of the token, but also compute its likelihood as a non-sentence boundary over the entire corpus. Hence, type-based approaches have the advantage of having the capacity to

Table 6 The performance of the introduced SBD methods. D denotes dataset

Method		Experiment 1			Experiment 2		
		D	R	M	D	R	M
Word	Stamatatos et al. (1999)	MG	99.4%	Acc.			
	Sadvilkar and Neumann (2020)	GRS	97.92%	Acc.			
	Treviso et al. (2017)	T-CTL	79%	F1	T-MCI	74%	F1
	Knoll et al. (2019)	MIMIC	98.6%	F1	FV	99.2%	F1
	Rudrapal et al. (2015)	BC	99.6%	Acc.	SMC	87.0%	Acc.
	Grefenstette and Tapanainen (1994)	BC	93.78%	Acc.			
	Reynar and Ratnaparkhi (1997)	BC	97.9%	Acc.	WSJ	98.8%	Acc.
	Schmid (2000)	BC	99.70%	Acc.	WSJ	99.62%	Acc.
	Kiss and Strunk (2002)				WSJ	99.05%	F1
	Kiss and Strunk (2006)	BC	98.89%	F1	WSJ	98.35%	F1
	Gillick (2009)	BC	99.64%	Acc.	WSJ	99.75%	Acc.
	Riley (1989)	BC	99.8%	Acc.			
	Wong and Chao (2010)	BC	99.98%	Acc.			
	Wong et al. (2014)	BC	99.81%	Acc.	WSJ	99.80%	Acc.
Syntax	Palmer and Hearst (1994)				WSJ	98.5%	Acc.
	Palmer and Hearst (1997)				WSJ	98.9%	Acc.
	Mikheev (2000)	BC	98.8%	Acc.	WSJ	99.2%	Acc.
	Mikheev (2002)	BC	99.72%	Acc.	WSJ	99.55%	Acc.
	Agarwal et al. (2005)	BC	97.7%	F1	WSJ	97.8%	F1
	Stevenson and Gaizauskas (2000)	T-News	76%	F1			
Prosody	Gotoh and Renals (2000)	T-R	70%	F1			
	Liu et al. (2004)	BN	48.61%	NIST	CTS	30.66%	NIST
	Liu et al. (2005)	BN	46.28%	NIST	CTS	29.30%	NIST
	Liu et al. (2006)	BN	49.57%	NIST	CTS	32.40%	NIST

R denotes result. M denotes measure. NIST is the NIST SU error rate

manage a large amount of unannotated data. Among type-based methods, Schmid (2000) implemented separate probability models for different scenarios. However, it requires the manual labor of listing such scenarios. Kiss and Strunk (2002, 2006) adverted this by framing SBD as a collocation detection problem of words and following periods. Gillick (2009) classified sentence boundary based on trigram context, only incorporating a type-based method for difficult instances, which is less computationally expensive. Treviso et al. (2017) suggested word embedding as a more effective alternative for *n*-gram and type-based approaches, bypassing feature engineering. Further on this direction, Knoll et al. (2019) utilized word-level and character-level neural networks to automatically extract morpholexical features.

Syntax-based systems are proposed based on the assumption that POS tags can provide context information where word-based approaches are lacking. The majority of syntax-based systems utilize syntactic features in conjunction with morpholexical ones to enhance the performance. Palmer and Hearst (1994, 1997) are the first to incorporate POS information, but wrongly assume that sentence boundary is a prerequisite for POS tagging. Notably, although these systems consist of a feedforward neural network, it is solely used for classification, not for feature extraction. Thus, they are not included in the neural category in Table 6. Mikheev (2000, 2002) solved the POS tagging dilemma, and developed a type-based method. Following this setup, Agarwal et al. (2005) used a token-based probabilistic model, whereas Stevenson and Gaizauskas (2000) tackled SBD in ASR with memory-based learning, which can be seen as type-based.

Lastly, since speech lacks the textual cues crucial to sentence boundary, prosody-based systems incorporate prosodic features for SBD in ASR. Gotoh and Renals (2000) and Liu et al. (2004, 2005) integrated prosodic features using various machine learning models. However, their systems are not adept in distinguishing SUs and sentences. Liu et al. (2006) confronted this weakness by tackling the imbalanced data distribution.

SBD for standard texts has already been well-studied. However, domain-specific SBD still remains a challenge, as some domains tend to use punctuations differently from general formal texts. Griffis et al. (2016) reviewed several off-the-shelf models on biomedical and clinical corpora. Their error analysis showed that the semicolons, colons, and newlines heavily used in clinical text are extremely error-prone. Additionally, periods used in unknown abbreviations, names, and numbers are a significant cause of error as well. Fatima and Mueller (2019) attempted to solve the task of SBD in financial domain via a machine learning approach and an unsupervised rule-based approach. Unfortunately, the former fails to produce acceptable results, whereas the performance of the latter is acceptable but still leaves a lot to be desired. Sanchez (2019) examined several off-the-shelf algorithms for SBD on legal texts. Similarly, the results are not ideal, indicating that there is still a lot of room for improvement for these existing approaches. Therefore, a robust SBD system that is capable of handling domain-specific corpora is called for.

Another challenging aspect is SBD in speech, especially SU boundary detection. Considering the lack of annotated resources in this domain, one possible direction for future work is semi-supervised learning. For instance, the machine can learn from annotated training samples that are manually labeled to precisely indicate SU boundaries, infer labels on the unannotated data, and fine-tune itself.

4 POS tagging

POS tagging is a fundamental task in NLP, which aims to label each word in a given text with its POS tag, e.g., noun, verb, adjective, etc. It is an upstream task that pre-processes the input texts to assist more complex NLP applications. Since the POS of a word can affect its meaning and polarity, POS tagging is important for downstream tasks e.g., word sense disambiguation (Taghipour and Ng 2015; Alva and Hegde 2016), information retrieval (Mahmood et al. 2017), sentiment analysis (Asghar et al. 2014; Mubarok et al. 2017), metaphor detection (Mao et al. 2021; Ge et al. 2022) and interpretation (Mao et al. 2018, 2022). Numerous studies have been done on POS tagging for a variety of languages (Shao et al. 2017; Nguyen et al. 2017; Kanakaraddi and Nandyal 2018; Darwish et al. 2018). Considering that the fundamental methodologies are similar across different languages, here we focus on introducing POS taggers in English.

English POS tagging is a well-studied problem. Following Church (1988), most early works utilized hand-crafted features, derived from the local contexts, e.g., rule-based learning (Brill 1995), memory-based learning (Daelemans et al. 1999b), and other statistical approaches, among which the MaxEnt framework (Ratnaparkhi 1996; Toutanvoa and Manning 2000; Curran and Clark 2003) and directed graphical models (Kupiec 1992; Brants

Table 7 Widely used corpora for POS tagging

Category	Dataset	Source	Reference
Formal	BC	the Brown Corpus	Francis and Kucera (1979)
	WSJ	Wall Street Journal	Marcus et al. (1993)
	Genia	Genia Biomedical corpus	Kim et al. (2003)
Social media	T-PoS	Twitter text corpus	Ritter et al. (2011)
	ARK	Twitter text corpus	Owoputi et al. (2013)
	NPS	Twitter text corpus	Forsyth (2007)

2000; McCallum et al. 2000) received the most attention. The performance of feature engineering approaches took a big leap with the novel CRF (Lafferty et al. 2001) and the perceptron algorithm (Collins 2002), which addresses the parameter estimation problem of MaxEnt models. Attempts at applying bidirectional inference to the task also achieved remarkable improvements, for instance the cyclic dependency network (Toutanova et al. 2003) and guided learning framework (Shen et al. 2007). With the recent surge of interest in deep learning, neural networks such as CNN (LeCun et al. 1998; Collobert et al. 2011), Recurrent Neural Network (RNN) and LSTM (Hochreiter and Schmidhuber 1997; Graves et al. 2013) have been applied to the field of POS tagging, freeing the task of POS tagging from hand-crafted feature set and yielding even more promising results. Subsequently, many studies extended the previous methods to semi-supervised learning to further boost the tagging accuracy (Clark et al. 2003; Suzuki and Isozaki 2008; Zhou et al. 2018). The performance of POS taggers are measured by accuracy, or in its other form, e.g., error rate.

In this paper, we first review the tagging schemas of POS tagging. Then, we divide the existing POS taggers into the following categories: feature engineering approaches, deep learning approaches, and semi-supervised learning approaches.

4.1 Tagging schemas

In this section, we introduce the most prevalent tagsets used in the field of POS tagging.

Penn Treebank POS tags The Penn Treebank POS tagset (Marcus et al. 1993) is the most widely used tagging paradigm. Derived from the Brown corpus, the Penn Treebank has since replaced it as the standard for POS tagging in English. Compared to its predecessor, the Penn Treebank offers a more fine-grained syntactic distinction, containing 36 POS categories in total. For instance, a base form of a verb is always tagged as VB in the Brown corpus, whereas the Penn Treebank differentiates it as VB (imperative or infinitive) or VBP (non-third person singular present tense) depending on the context.

However, as the need for multilingual and cross-lingual POS induction arises, the granular differentiation of the Penn Treebank becomes a drawback, since many languages do not entirely follow the grammatical structure of English.

Universal POS tags To address the weakness of the Penn Treebank and facilitate future research, Petrov et al. (2011) proposed a tagset with coarser syntactic POS categories based on the observed commonalities across most languages. The Universal tagset contains 12 universal POS categories. In addition, they also created a mapping from 25 Penn Treebank POS categories to their tagset. When used in conjunction with the Penn Treebank, the Universal tagset and mappings are able to account for 22 different languages.

The commonly used corpora for POS tagging (in English) are listed in Table 7.

4.2 Feature engineering approach

Feature engineering approaches, as the name suggests, rely on hand-crafted feature templates to determine the POS tag for a word. Some commonly used features are lexical, contextual, and morphological. To avoid confusion, here we define lexical features as the frequencies of the observed POS tags in the training data, contextual features as the surrounding words and their predicted POS tags if available, morphological features as affixes, and existence of numerals, hyphens, capitalization, etc. The taggers utilize such features via rule-based or machine learning algorithms. Notably, graphical models such as HMM and CRF gain most attention because of their proficiency at capturing sequential dependency.

Church (1988) was the first to determine that features from two or less of the nearby tokens are significantly informative to predicting the POS tag of a given token. For example, the word *bear* can be a verb or a noun, but if it is observed to follow a determiner such as *the*, then the tagger can label *bear* in the word sequence *the bear* as noun. The tagger takes a lexicon-based approach - for each word in the corpus, their most frequent POS tags and the corresponding lexical probabilities are stored in a lookup table. The tagger also computes the contextual probability, which is the probability of observing a POS tag given the two POS tags following it. Then, given a sentence, the goal is to search for the tag sequence that optimizes the lexical and contextual probabilities.

Brill (1995) proposed a POS tagger with transformation-based learning. Initially, the tagger assigns each word the most likely POS tag according to the training corpus. When the prediction is incorrect, the tagger tries another transformation from the transformational rule templates, which is derived from the three preceding words, the three following words, and their POS tags. The learning procedure stops when no more transformation can be found to reduces the errors. This method is simple and effective, but unfortunately require a long training time. To address this problem, Ngai and Florian (2001) optimized the model by incorporating good and bad counts for each transformational rule, which avert repetition in the learning procedure. This method successfully reduces the training time while maintaining the accuracy.

Daelemans et al. (1999b) introduced a memory-based approach to POS tagging. Examples in training set are represented as a feature vector with an associated tag category. For each test data point, its similarity to all examples in the memory is computed. The category of the most similar instances is chosen as the predicted category for the test data points.

Abney et al. (1999) applied boosting to POS tagging. Boosting algorithm is similar to transformation-based learning discussed above, where the model combines template rules to produce the most accurate classification rule. They proposed two methods to deal with the multi-class problem. First they applied the AdaBoost.MH algorithm (Schapire and Singer 1999), where each possible class is paired with the given word and assigned a binary label as a derived problem, which is then solved by binary AdaBoost. AdaBoost.MH is memory-consuming, therefore they proposed a novel AdaBoost.MI algorithm, which uses binary AdaBoost to train separate binary classifiers for each class, and combines their output by choosing the class with most confidence. Unlike AdaBoost.MH, here the predictions are selected independently for each class. The boosting approach is shown to be more accurate than transformation-based learning.

Nakagawa et al. (2001) employed SVM to specifically solve the unknown word problem in POS tagging. In this method, binary SVM classifiers are created for each POS tag based on the training corpus, which are then used to predict the POS tags of unknown words. Subsequently, Giménez and Marquez (2004a) and Giménez and Màrquez (2004b) proposed SVMTool, which extends the binary SVM to cover multi-class classification. SVM classifiers are created for each POS tag that contains ambiguous lexical items. When labeling a word, the most confident prediction among all the SVM classifiers is selected.

Ratnaparkhi (1996) first successfully applied MaxEnt framework to POS tagging A set of binary features is paired with every possible output label. Each feature-label pair is assigned a weight which is trained to maximize the entropy. When predicting the POS tag of a word, these weights are multiplied by the corresponding feature to estimate the probability of a label. The label with the highest probability is selected. Toutanvoa and Manning (2000) improved this method by modifying the set of features. They removed the lexical features derived from the preceding words, adding other hand-crafted features derived from a larger context window, which alleviates the proper noun problem and the ambiguous word form distinction problem. Curran and Clark (2003) proposed a re-implementation of the MaxEnt tagger (Ratnaparkhi 1996) using Generalized Iterative Scaling (GIS) estimation algorithm and model smoothing technique. Their method proves to be much faster in training and predicting than the original MaxEnt tagger.

Directed graphical model is another well-studied statistical approach to POS tagging. Kupiec (1992) proposed a POS tagger based on HMM. HMM model follows the first order Markov assumption where the current state is only conditioned on previous states, and the current observation is only conditioned on the current state. For POS tagging, the observations are that the words in a given input sentence x, and the states are the corresponding POS tags y. The objective is to estimate the joint distribution $P(x, y)$ by computing the transition probability and the emission probability. The former links the current state with the previous one, whereas the latter represents how likely a word is observed under a given label. The transition matrix and emission matrix are estimated from frequency counts using a tagged training dataset.

Brants (2000) proposed a Trigram'n'Tags (TnT) POS tagger based on second order Markov model combined with a smoothing technique. It is a generative model, similar to HMM. The states in the model represent tags. The outputs represent words. The transition probabilities depend on the states, whereas the output probabilities only depend on the most recent class. The trigram probability is computed using a smoothing paradigm that is a linear interpolation of unigrams, bigrams and trigrams. This POS tagger requires less time to train while achieving similar accuracy as a MaxEnt tagger (Ratnaparkhi 1996).

McCallum et al. (2000) presented Maxium Entropy Markov model (MEMM), a novel Markovian sequence model combining HMM and MaxEnt framework. HMM tagger needs to enumerate all possible observation sequences to define the joint probability $P(x, y)$. Additionally, the inference is intractable. Therefore, it is not cost-effective for HMM to represent multiple interacting features or long-range dependencies of the observations. Conditional probabilistic sequence model such as MEMM is a good alternative to alleviate this difficulty, since it defines the probability of possible label sequence given an observation sequence, i.e., $P(y \mid x)$. It also allows observations to be represented as arbitrary overlapping features. The MEMM uses the MaxEnt framework to fit a set of exponential models, where the probability of a state depends on the observations and the previous state.

Although MaxEnt models are effective and widely used for sequence labeling tasks, their parameter estimation methods have limitations. In MEMM, the transitions leaving a given state compete locally instead of globally, which causes a bias towards states with fewer outgoing transitions. Thus, it cannot accurately represent dependencies between consecutive states. Different methods was proposed to solve this label bias problem.

Collins (2002) proposed a perceptron-based (Rosenblatt 1958) parameter estimation algorithm for global learning. The tagger predicts a sequence of tags from left to right, where each tag is determined based on the input sequence and the previous tags, also known as history. The tagging task is then represented as a feature-vector representation of history-tag pairs. The history-tag pairs are seen as local representations, which sums up to the global representation of the input sequence. A score function is defined based on the global representation, which is used to estimate the parameters corresponding to each history. The highest scoring tag sequence for the given sentence is found via Viterbi algorithm. In the training phase, if the highest scoring sequence is incorrect, the parameters will be updated using simple addition and subtraction.

Another more widely-used alternative to MaxEnt is CRF, which is a novel method as opposed to the sequential classification approaches mentioned above. Lafferty et al. (2001) is the first to introduce CRF to sequence labeling. CRF is a discriminative linear-chain graphical model. It uses a single exponential model for the joint probability of the entire state sequence given an observation sequence. Thus, the weights of different features at different states can be traded off against each other. The graph model can be expressed as $G = (V, E)$, where V denotes the nodes, i.e., states and observations, and E denotes the undirected edges that represent the dependencies between variables. The features are defined over each edge in the graph. $x = (x_1, x_2, \dots, x_n)$ denotes the input sequence. $y = (y_1, y_2, \dots, y_n)$ denotes the output sequence. Then each y_i is dependent on x and y_{i-1}, represented by a set of binary feature functions $\boldsymbol{F}$. The joint probability is thus computed as

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\lambda \cdot \boldsymbol{F}(y, x)),$$

where Z is the normalization factor, and λ denotes weights for each feature function.

Rush et al. (2012) proposed a POS tagger using Markov random fields (MRFs) to model global constraints between sentences, which alleviates the accuracy problem when labeled dataset is limited or out-of-domain. MRF is an undirected graph model $G = (V, E)$, where V are nodes for every word in the given sentence. An index set is constructed to include all valid label assignments in the corpus, which acts as inter-sentence constrains. The objective is to find the best sentence-level label assignments with regards to V and E that are consistent with elements in the index set, incorporating corpus-level information to augment sentence-level labeling.

Although undirected graphical models such as CRF avert the label bias problem, they have a few disadvantages against the sequential classification method. On one hand, CRF is less efficient to train because they need to perform Viterbi algorithm over the entire sentence in each iteration. On the other hand, sequential classification method leaves room to employ a variety of machine learning algorithms as local classifiers. Therefore, many works focus on improving the performance of sequential classification methods by enriching the information for the local classifiers. Since the previous directed graphical taggers take an unidirectional approach, one popular way of enhancing features is utilizing future tags via bidirectional networks.

Toutanova et al. (2003) was the first to introduce bidirectional inference for POS tagging. They proposed a cyclic dependency network with a series of local conditional log-linear models to exploit information from both directions explicitly. Each node in the network represents a random variable with a corresponding local conditional probability model that considers the source variables from all incoming arcs. The tagger finds the sequence that maximizes the score via Viterbi algorithm, similar to previous MaxEnt and HMM models. The only difference between bidirectional inference and an unidirectional graphical model such as HMM is that, when the Markov window is at the time step i, the score it receives is $P(t_{i-1} \mid t_i, t_{i-2}, w_{i-1})$ instead of $P(t_i \mid t_{i-1}, t_{i-2}, w_i)$, t and w being output tag and input word respectively. Their model, however, suffers from collusion problem where the model lock onto conditionally consistent but jointly unlikely sequences. This is because the local classifiers encounter double-counting problem when using the information from future tags.

In order to avert this problem, Tsuruoka and Tsujii (2005) proposed an alternative bidirectional inference algorithm with an easiest-first strategy. The label sequence of a given sentence is the product of local probabilities. The proposed inference method is to consider all of the possible decomposition structures and choose the optimal structure to predict label sequence. The paper also proposed a more efficient alternative to bidirectional decoding algorithm, which adopts the easiest-first strategy. Instead of enumerating all the possible decompositions, the tagger tags the easiest word at each step, and repeating the procedure until all the words are tagged. To pick the easiest word, the appropriate local MaxEnt classifier is selected according to the availability of the neighboring labels, and used to output the probabilities. The word with the highest probability is deemed as the easiest word for the current step. Their bidirectional inference method is proven to be able to find the highest probability sequence with similar performance but lower computational complexity.

Shen et al. (2007) proposed a novel guided learning framework for bidirectional inference. Unlike the easiest-first strategy which only uses heuristic rule to determine the order of inference, their approach incorporates the selection of inference order into the training of the MaxEnt classifier for individual token labeling, combining the two into a single learning task. Specifically, a sub-sequence of the input sentence is called a span. Each span is associated with one or more hypotheses, which are started and grown via labeling actions. The tagger initializes and maintains a set P of accepted spans and a set Q of candidate spans. It repeatedly selects a candidate span from Q whose action score of its top hypothesis is the highest,moving it to P, until a span covering the whole input sentence is added to P from Q. For training, the tagger uses guided learning to learn the weight of action score. If the top hypothesis of a selected span is compatible with the gold standard, then the candidate span is accepted. Otherwise, similar to perceptron algorithm (Collins 2002), the weight is updated by rewarding the feature weights of the gold standard action and punishing the feature weights of the action of the top hypothesis. Then all the existing spans in Q are removed and replaced with new hypotheses for all the possible spans generated based on the context spans in P. This allows the tagger to simultaneously learn the individual classification and the inference order selection.

Ma et al. (2013) proposed an easy-first POS tagger with beam search. The easy-first POS tagger enumerates all possible word-tag pairs, choosing the most confident one to label according to the score function, marking the word as processed. Then the tagger recomputes the scores of the unprocessed words based on the local context, repeating the selection procedure until all the words are marked as processed. For the easy-first tagging with beam search, a set of labeling action sequences is maintained and grows via beam search. At each step, the sequences in the beam β are expanded in all possible ways, and the top expanded sequences within the beam width are selected into β. The trainable weight vector in a score function is learned through perceptron-based global learning similar to the previously mentioned guided learning framework (Shen et al. 2007), however its performance on the WSJ dataset is not as good as the latter.

4.3 Deep learning approach

So far the above-mentioned methods are all dependent on hand-crafted features sets. With the development of deep learning in NLP, the application of neural networks to POS tagging makes it possible to avoid feature engineering and further improve the tagging performance. Notably, neural network models are widely-employed to automatically capture character-level patterns, which have to be modeled with morphological features in previous taggers, e.g., suffix, capitalization, presence of numerals, etc.

Collobert et al. (2011) proposed a window approach for sequence labeling tasks, which assumes the tag of a word is mainly dependent on its neighboring words. Hence it considers a fixed size window of words around the current word as local features. Given an input sentence, the tagger passes it through a lookup table layer. The resulting sequence of representations is fed into the convolutional layer, which can be regarded as a feed-forward neural network with L layers, to extract local features. That is, a concatenation of the word vectors in the focal window is inputted to L linear layers to perform affine transformations over their inputs. Finally, a softmax layer computes the probabilities for each labels given the output of the L-th layer. The POS tagger is trained with word-level log-likelihood, also more commonly known as cross-entropy.

Dos Santos and Zadrozny (2014a) used deep neural network to learn the character-level representations, and combined them with the corresponding word representations to perform POS tagging. Prior to their work, the morphological or other intra-word information is given to the tagger via hand-crafted features. To reduce human effort, CharWNN is proposed as an extension of the previously introduced convolutional architecture (Collobert et al. 2011). CharWNN uses the convolutional layer to extract features from the input words and generates their character-level embedding at tagging time. The character-level embeddings are concatenated with word-level embeddings as word representations. Taking the window approach, a fixed window size of word representations in the input sentence are concatenated into a vector, which are then fed into two linear neural network layers to compute the scores. The tagger also incorporates the transition score (Collobert et al. 2011), which is introduced in detail in the text chunking section, in order to capture the structural information from the sentence.

Wang et al. (2015) proposed a BLSTM-RNN model for POS tagging. The model first implements a linear layer as a lookup table to produce word embeddings, which are fed into a bidirectional LSTM layer and then a softmax layer to output the scores of tags. They also introduced a novel method to train word embedding on unlabeled data, where BLSTM-RNN takes a sentence with some words replaced by randomly chosen words as input, tagging the words in the sentence as correct or incorrect. Thus the lookup table in BLSTM-RNN is trained to minimize the binary classification error.

Ma and Hovy (2016) introduced a Bi-LSTM-CNN-CRF model, which utilizes both word-level and character-level representations automatically, requiring no task-specific resources, feature engineering or data pre-processing. First, a CNN layer (Chiu and Nichols 2016) is applied to encode character-level information of a word into its character-level representation. A dropout layer is applied before character embeddings are feed into CNN. Then, the model concatenates the character-level representations and word embeddings, feeding them into Bi-LSTM to model context information of each word. A dropout layer is also applied to the output vectors. Lastly, the output vector of Bi-LSTM is fed to a CRF layer to decode labels for the whole sentence.

Akbik et al. (2018) proposed Flair, a contextual string embedding for sequence labeling tasks. The Flair embedding is learned using a LSTM-based language model over sequences of characters instead of words. For optimal performance, the Flair embedding is stacked together with pre-trained static word embedding GloVe (Pennington et al. 2014) and a task-trained character embedding learned by LSTM. The final embedding is passed onto a standard BiLSTM-CRF architecture to acquire the output label sequence.

Zhao et al. (2019) proposed a deep CNN architecture called Deep Gated Dual Path CNN (GatedDualCNN) for sequence labeling. The model first uses a CharCNN to extract character-level representations, which are then concatenated with the word embeddings and fed into a 1-D convolution (Conv1D) layer followed by rectified linear unit (ReLU) and batch normalization (BN) to get the inital hidden states. Thereafter, in order to stack up more convolutional layers while averting the vanishing gradient problem, the paper incorporated gate blocks, residual connection, and dense connection. The core component of a gate block is the gated linear unite (GLU) (Dauphin et al. 2017), whose output is processed by a Conv1D layer with ReLU and BN to produce the successive hidden state. To encourage feature re-usage between gate blocks, residual connection is introduced to bypass the non-linear transformation in the gate block. On the other hand, dense connection serves the purpose of new feature exploration in a dense path. To combine these two connections, the model uses a dual path, where hidden state produced by each block is split row-wise, then fed into the residual path and the dense path respectively. The outputs are concatenated as the input of the next block. The final hidden state is then passed on to a CRF layer to decode the best sequence of tags.

Yang et al. (2017) introduced transfer learning for deep hierarchical RNN POS tagger to alleviate the out-of-domain problem. The base model uses a character-level GRU (Cho et al. 2014) to obtain character embeddings, which are concatenated with word embeddings and passed on to a word-level GRU and a CRF layer to predict the tag sequence. They described three different transfer learning architectures for this base model. Transfer model T-A, which shares all the model parameters and feature representations between domains, is used for cross-domain transfer where label mapping is possible. It only performs a label mapping step on top of the base's CRF layer. If the two domains have disparate label sets, then transfer model T-B learns a separate CRF layer for each tasks while sharing parameters in other layers. For cross-lingual transfer, model T-C only shares the parameters and representations in the character-level GRU, keeping two separate word-level GRU and CRF layers for the source task and the target task. The paper experiments with transferring from chunking and Name Entity Recognition (NER) to standard POS tagging on WSJ, and also tests transfer learning from WSJ to Genia Biomedical corpus (Kim et al. 2003) and Twitter corpus T-PoS.

Similarly, Meftah and Semmar (2018) presented a transfer-learning-based end-to-end neural model. Their base model uses a CNN layer to extract character embedding, a GRU layer to compute hidden states, and a fully-connected layer and softmax layer to output the scores for tags. Two transfer learning architectures are proposed based on this neural network. For cross-domain transfer, they used a parent network for source data and a child network for target data. The parent network is trained on annotated out-of-domain data, namely WSJ, whose parameters are transferred to the child network. Then the child network is fine-tuned through training on labeled Twitter datasets. For cross-task transfer, the parent network and the child network shares a set of parameters, jointly optimizing the two tasks, while maintaining separate task-specific parameters that are trained on the corresponding task. The task selected for the parent network is NER.

4.4 Semi-supervised approach

As researchers have easy access to large amounts of unlabeled texts online, it is a natural progression for POS taggers to utilize unlabeled data. In addition, POS taggers trained from annotated dataset generally achieve high accuracy on in-domain data, whereas their performance drops on out-of-domain data such as social media posts and conversational texts. Consequently, there have been many attempts to use unsupervised or semi-supervised learning to alleviate the OOV problem that is largely responsible for the fall in tagging accuracy.

Unsupervised learning POS taggers mostly take the word clustering approach. For example, Clark (2003) described an unsupervised POS tagger that incorporates morphological information into a distributional clustering algorithm to sort unlabeled text into lexical classes. Biemann (2006) introduced a graph-based word clustering method based on context similarity of high frequency words and log-likelihood of lower frequency words. Owoputi et al. (2013) proposed an unsupervised MEMM tagger for word clustering on Twitter conversations.

Semi-supervised learning (SSL) approach, on the other hand, utilizes both small amounts of labeled data and relatively large amounts of unlabeled data for training. SSL is more scalable than unsupervised learning. It can be applied to POS tagging to solve the out-of-domain problem and improve accuracy.

Clark et al. (2003) proposed bootstrapping POS tagger using co-training with unlabeled data. The main idea is to co-train a TNT tagger (Brants 2000) and a MaxEnt tagger (Curran and Clark 2003), using the output from one as additional labeled data for the other. The model first learns two separate classifiers for each view of the task via a small amount of labeled seed data. Then, each classifier incrementally labels a subset of unlabeled data for the other to use as the new training dataset. Results show that when using little labeled training data and a much larger amount of unlabeled data, the accuracy of POS tagger improves. However, the performance drops when the amount of labeled training data increases.

Ando and Zhang (2005) proposed a semi-supervised multitask learning framework. The objective is to learn from unsupervised tasks, and transfer to the target supervised task. The framework consists of multiple task classifiers, each made of two alternatively optimized feature functions. One is task-specific and the other is trained to map into a low dimensional subspace common across all tasks. For POS tagging, two unsupervised auxiliary tasks are used. The first task is word prediction, which predicts the word at the current position. The second task is Top-2, which predicts the top-2 choices of the POS tagger trained with labeled data.

Toutanova and Johnson (2007) presented a Bayesian Latent Dirichlet Allocation (LDA) model for semi-supervised POS tagging. Compared to previous semi-supervised approaches, the tagger is not given any labeled data, but a dictionary that constrains the possible tags of some words. The LDA-based model is a generative model that uses only observed context features to predict the tags of words. They incorporated a sparse prior on the distribution over tags for each word, and employed a Bayesian approach that maintains a distribution over parameters. The semi-supervised LDA is enabled by explicitly modeling ambiguity classes obtained from the dictionary constrain.

Spoustová et al. (2009) extended the average perceptron algorithm (Collins 2002) for semi-supervised learning. Specifically, they used an ensemble of taggers (Brill and Wu 1998; van Halteren et al. 2001) to pre-tag a large unannotated corpus, which is then combined with labeled dataset and fed into an average perceptron POS tagger. Furthermore, they observed that it is beneficial to feed the tagger with different chunks of unsupervised data at each training iteration. Hence, they explored three types of selection mechanisms, namely, sequential chopping, random selection without replacements, and random selection with replacements. Based on the properties of the average perceptron algorithm, the selected chunk of unannotated data is inputted after the labeled dataset. Results show that the semi-supervised method indeed outperforms supervised ones, and the three mechanisms contribute identically for English POS tagging.

Suzuki and Isozaki (2008) introduced a semi-supervised extension of CRF (Lafferty et al. 2001) that combines supervised and unsupervised probability models via a parameter estimation called Maximum Discriminant Functions sum (MDF). For a given input sequence x and its corresponding output y, C denotes the set of cliques in an undirected graphical model. y^c is the feature function and the output from the associated clique c. Then, similar to standard CRF, the conditional probability for y is

$$P(y \mid x) = \frac{1}{Z(x)} \prod_{c \in C} \exp(\lambda \cdot \mathbf{F}^c(y^c, x)),$$

where $Z(x)$ is the normalization factor, $\mathbf{F}^c$ is the set of feature function for the corresponding clique c, and λ is the function weights. In this SSL-based architecture, the feature functions for clique c are the concatenation of $\mathbf{F}^c$ and the log likelihood of all the joint probability models. A set of model parameters θ is introduced to weight the joint models, which is estimated using unlabeled data via MDF.

Subramanya et al. (2010) described another CRF-based algorithm for semi-supervised POS tagger. For graph construction, they used local sequence contexts as graph vertices V, which consists of a set V_l of n-grams that occur in the labeled data and a set V_u in the unlabeled data. The graph is built over types rather than tokens via a symmetric similarity function, thus named similarity graph. This similarity graph is used as a smoothness regularizer to train CRF in a semi-supervised manner. Specifically, given a set of CRF parameters, the tagger first computes marginals over tokens in the unlabeled data, which are then aggregated to marginals over types and used to initialize the graph label distributions. After running label propagation, the posteriors from the graph are used to smooth the state posteriors. Subsequently, the unlabeled data is decoded using Viterbi algorithm to produce a set of automatic annotations, which are combined with the labeled data to retrain the CRF through supervised learning. The procedure is repeated until convergence. They used WSJ as labeled source domain training data, and the QuestionBank (Judge et al. 2006) as test data. The unlabeled data are collected from Internet search queries in similar forms to the QuestionBank. Experiments show that their proposed algorithm indeed outperforms supervised CRF in other domains.

Søgaard (2010) introduced stacked learning as a way to reduce POS tagging to a classification task, thus simplifying semi-supervised training. The stacking approach here is to combine SVMTool (Giménez and Màrquez 2004b) and an unsupervised tagger (Biemann 2006) into a single-end classifier, where the former predicts the POS tag of a given word and the latter sorts the word into a word cluster. Semi-supervised learning is achieved by tri-training with disagreement. Firstly, three classifiers of the same learning algorithm mentioned above are trained on three bootstrap samples of the labeled dataset, which ensures that the classifiers are diverse. Then, a data point in the unlabeled dataset is labeled for classifier c_1, if and only if the other two agree on its label assignment but c_1 disagrees, which strengthens the weakness of the classifier without skewing the labeled data by easy data points. This labeling process is repeated until the classifiers no longer change. Subsequently, the three classifiers are integrated by majority voting.

Zhou et al. (2018) proposed a weakly supervised sequence tagging model with ECOC (Error-Correcting Output Codes) that can learn to predict the POS tag for a given word in a context, given a dictionary of words with their possible tags. Most approaches prior to this paper are based on disambiguation, such as CRF and HMM, which suffers from the negative effects of false positive tags as the size of possible tags increases. The POS tagger is trained and tested based on constrained ECOC (Dietterich and Bakiri 1994). First, a unique L-bit vector is assigned to each tag. The set of bit-vectors is regarded as a coding matrix, where each row represents a codeword, e.g., class, and each column specifies a dichotomy over the tag space to learn a binary classifier. In the encoding stage, for each column of the coding matrix, a binary classifier is built based on binary training examples derived from the dictionary of the words with their possible tags. In the decoding stage, the codeword of an unlabeled test instance is generated by concatenating the predictive output of the L binary classifiers. The predicted instance is the class with the closest codeword according to hamming distance or Euclidean distance. Thus, the proposed model which not only treats the set of possible tags as an entirety without resorting to disambiguation procedure, but also needs no manual intervention for feature engineering.

Gui et al. (2017) proposed a Target Preserved Adversarial Neural Network (TPANN) for POS tagging on Twitter. WSJ is used as the labeled out-of-domain data, T-PoS, ARK, and NPS as labeled in-domain data, and tweets collected via Twitter API as unlabeled in-domain data. The objective is to learn common features between resource-rich domain and target domain while preserving some domain-specific features of the target domain. TPANN first extracts character embedding features via CNN and concatenates them to word embedding as input. The hidden states are produced by a Bi-LSTM layer. Subsequently, the hidden states are transferred to a POS tagging classifier and a domain discriminator, which are both standard feed-forward networks with a softmax layer. The POS tagging classifier maps the hidden states to their labels, whereas the domain discriminator maps the same hidden states to the domain labels so as to make the input features domain-invariant. By training this adversarial network, common features can be obtained, but some domain-specific features are weakened. Thus, the paper introduced a domain-specific auto-encoder to reconstruct target domain data. Specifically, at the Twitter decoder side, the hidden state h_t is computed with $h_t = LSTM([h_0 \oplus z_{t-1}], h_{t-1})$, where h_0 is the last hidden state of the Bi-LSTM layer, $\oplus$ denotes the concatenation operation, and z_{t-1} is computed from h_{t-1} using a multiple perceptron function. In this way, the auto-encoder counteracts the adversarial network's tendency to erase target domain features by optimizing the common representation to be informative on the target domain data.

4.5 Summary

POS tagging is a well-researched problem in the field of NLP. The existing POS tagging models can be divided into feature engineering approaches and deep learning approaches. Nonetheless, to address the growing need for domain adaptable POS taggers, we introduce another category to introduce semi-supervised methods.

Early feature engineering methods predict the POS tag of a word based on its local n-gram context with various machine learning techniques, following Church (1988). Graphical models are commonly used for sequence labeling. Generative graphical models (Kupiec 1992; Brants 2000) estimate the joint distribution based on the explicit dependency between the states and the observations, and thus difficult to accommodate context features. Directed discriminative model (McCallum et al. 2000) addresses this weakness by modeling the conditional probability based on the dependencies between adjacent states and the observation sequence, while it suffers from the label bias problem. CRF (Lafferty et al. 2001) is able to solve the limitations above. It is an undirected discriminative graphical model that is able to leverage decisions globally. However, it is disadvantageous in computational efficiency. Collins (2002) proposed an alternative that enables global learning by modifying the parameter estimation process of directed graphical model. Others alleviate the local limitation by introducing bidirectional sequence classification (Toutanova et al. 2003; Tsuruoka and Tsujii 2005; Shen et al. 2007). These methods fallen out of favor in recent years, because it is not as adaptable to deep learning as CRF. MRF (Rush et al. 2012) is an undirected skip-chain graphical model that makes global decision not only on sentence-level but also corpus-level. Computationally it is as costly as CRF.

For deep learning, early methods take the window approach (Collobert et al. 2011; Dos Santos and Zadrozny 2014a) to extract contextual information. With the advancement of neural networks in NLP, it is replaced by RNN and RNN variants Wang et al. (2015), which is better suited for processing text. The most common approach is the BiLSTM-CRF architecture, with improvement achieved through exploration of embeddings (Ma and Hovy 2016; Akbik et al. 2018). Zhao et al. (2019) proposed a CNN-based model that addresses the vanishing gradient problem of deep CNN, which outperforms BiLSTM-CRF when using the same embeddings. For transfer learning on cross-domain POS tagging, Meftah and Semmar (2018) relied on hard parameter sharing, whereas Yang et al. (2017) utilized hard and soft parameter sharing on different scenarios.

For SSL, Clark et al. (2003), Søgaard (2010), and Spoustová et al. (2009) take the ensemble approaches. Clark et al. (2003) and Søgaard (2010) co-trained multiple POS taggers, whereas Søgaard (2010) introduced a tri-training strategy to reduce errors. Spoustová et al. (2009) used ensemble to label unannotated data for a specific tagger, which risks error propagation. Ando and Zhang (2005) jointly trained two unsupervised auxiliary tasks with supervised POS tagging. Toutanova and Johnson (2007) and Zhou et al. (2018) used dictionary constrains instead of labeled data. The former method was based on clustering, thus difficult to evaluate. The latter used constrained ECOC to solve the false positive tag problem of disambiguation-based methods. Suzuki and Isozaki (2008) and Subramanya et al. (2010) extended CRF to SSL by introducing cliques of states and similarity graph respectively. The former focused more on incorporating unlabeled data, whereas the latter targeted cross-domain scenarios. Compared to other cross-domain taggers, TPANN (Gui et al. 2017) is advantageous in utilizing a large amount of unlabeled data through a domain-specific auto-encoder.

Table 8 illustrates a summary of the features and properties of all the discussed models. For feature engineering methods, there are mainly three types of features. First, lexical features are the observed POS tags of the focal word in the training corpus and their corresponding frequencies, or less commonly, other frequency counts relating to the word, e.g., capitalized form frequency (Toutanvoa and Manning 2000). Second, contextual features are the words surrounding the focal word and their predicted POS tags if available. Third, morphological features are the focal word's suffix, prefix, and existence of numerals and special symbols. For some taggers less adept at handling overlapping features, only rare words in the corpus are given such morphological features. Thus we mark out the methods using rare word distinction. For methods using neural networks, on the another hand, such features can be automatically extracted via character-level encoding. Additionally,

Table 8 The comparison between different POS tagging methods

Method		Lexical	Contextual	Morphological	Rare Word	Character	Global	Neural
Feature engineering	Church (1988)	✓	✓					
	Brill (1995)		✓					
	Ngai and Florian (2001)		✓					
	Daelemans et al. (1999b)	✓	✓					
	Abney et al. (1999)	✓	✓	✓				
	Nakagawa et al. (2001)		✓	✓				
	Giménez and Màrquez (2004b)		✓	✓				
	Ratnaparkhi (1996)		✓	✓	✓			
	Toutanvoa and Manning (2000)	✓	✓	✓	✓			
	Curran and Clark (2003)		✓	✓		✓		
	McCallum et al. (2000)		✓	✓				
	Kupiec (1992)	✓	✓	✓		✓	✓	
	Brants (2000)		✓	✓	✓		✓	
	Lafferty et al. (2001)		✓	✓			✓	
	Collins (2002)		✓				✓	
	Rush et al. (2012)		✓	✓			✓	
	Toutanova et al. (2003)		✓	✓			✓	
	Tsuruoka and Tsujii (2005)		✓	✓			✓	
	Shen et al. (2007)		✓	✓			✓	
	Ma et al. (2013)		✓	✓			✓	

Table 8 (continued)

Method		Lexical	Contextual	Morphological	Rare Word	Character	Global	Neural
Deep learning	Collobert et al. (2011)					✓	✓	
	Dos Santos and Zadrozny (2014a)					✓	✓	✓
	Wang et al. (2015)						✓	✓
	Ma and Hovy (2016)					✓	✓	✓
	Akbik et al. (2018)					✓	✓	✓
	Zhao et al. (2019)					✓	✓	✓
	Yang et al. (2017)					✓	✓	✓
	Meftah and Semmar (2018)					✓	✓	✓
Semi-supervised	Clark et al. (2003)		✓	✓	✓			
	Ando and Zhang (2005)	✓	✓					
	Toutanova and Johnson (2007)		✓	✓				
	Spoustová et al. (2009)		✓	✓			✓	
	Suzuki and Isozaki (2008)		✓	✓			✓	
	Subramanya et al. (2010)		✓	✓			✓	
	Søgaard (2010)	✓	✓					
	Zhou et al. (2018)		✓				✓	✓
	Gui et al. (2017)					✓	✓	✓

Table 9 The performance of the introduced POS methods

Method		Experiment		
		D	R	M
FE	Brill (1995)	WSJ	96.6%	Acc.
	Ngai and Florian (2001)	WSJ	96.61%	Acc.
	Daelemans et al. (1999b)	WSJ	96.6%	Acc.
	Abney et al. (1999)	WSJ	96.68%	Acc.
	Nakagawa et al. (2001)	WSJ	97.1%	Acc.
	Giménez and Màrquez (2004b)	WSJ	97.05%	Acc.
	Ratnaparkhi (1996)	WSJ	96.63%	Acc.
	Toutanvoa and Manning (2000)	WSJ	96.86%	Acc.
	Curran and Clark (2003)	WSJ	97.27%	Acc.
	Kupiec (1992)	WSJ	95.7%	Acc.
	Brants (2000)	WSJ	96.7%	Acc.
	Collins (2002)	WSJ	97.11%	Acc.
	Lafferty et al. (2001)	WSJ	95.73%	Acc.
	Rush et al. (2012)	WSJ	91.98%	Acc.
	Toutanova et al. (2003)	WSJ	97.24%	Acc.
	Tsuruoka and Tsujii (2005)	WSJ	97.24%	Acc
	Shen et al. (2007)	WSJ	97.33%	Acc.
	Ma et al. (2013)	WSJ	97.28%	Acc.
DL	Collobert et al. (2011)	WSJ	97.37%	Acc.
	Dos Santos and Zadrozny (2014a)	WSJ	97.47%	Acc.
	Wang et al. (2015)	WSJ	97.40%	Acc.
	Ma and Hovy (2016)	WSJ	97.55%	Acc.
	Akbik et al. (2018)	WSJ	97.85%	Acc.
	Zhao et al. (2019)	WSJ	97.59%	Acc.
	Yang et al. (2017)	WSJ	97.55%	Acc.
	Yang et al. (2017)	Genia	92.62%	Acc.
	Yang et al. (2017)	T-PoS	83.65%	Acc.
	Meftah and Semmar (2018)	T-PoS	90.90%	Acc.
	Meftah and Semmar (2018)	ARK	92.01%	Acc.
	Meftah and Semmar (2018)	NPS	93.20%	Acc.
Semi	Clark et al. (2003)	WSJ	409	Perplexity
	Ando and Zhang (2005)	BC	93.1%	Acc.
	Toutanova and Johnson (2007)	WSJ	93.4%	Acc.
	Spoustová et al. (2009)	WSJ	97.44%	Acc.
	Suzuki and Isozaki (2008)	WSJ	97.35%	Acc.
	Subramanya et al. (2010)	QB	86.8%	Acc.
	Søgaard (2010)	WSJ	97.27%	Acc.
	Zhou et al. (2018)	WSJ	92.91%	Acc.
	Gui et al. (2017)	T-PoS	90.92%	Acc.
	Gui et al. (2017)	ARK	92.80%	Acc.
	Gui et al. (2017)	NPS	94.10%	Acc.

D denotes dataset. R denotes result. M denotes measure. FE stands for feature engineering. DL stands for deep learning. Semi stands for semi-supervised. QB is QuestionBank some methods determine the optimal output sequence by making the best local decision, whereas others are able to leverage predictions at different positions (see the global column in Table 8).

Table 9 shows a comparison of the accuracy of the introduced POS taggers. Currently, most POS tagger yield stable and reliable performance on the WSJ dataset from the Penn Treebank, with accuracy slightly above 97%. Akbik et al. (2018) obtained the state-of-the-art result of 97.85%. On Twitter text corpa, Gui et al. (2017) consistently achieved the best performance.

Manning (2011) performed an error analysis and categorized the common errors from POS taggers into 7 classes. Results indicate that, a proportion of errors due to lexicon gap and unknown words can be addressed by semi-supervised methods. The errors caused by inconsistent or faulty gold standards can be fixed by correcting the WSJ dataset, for which he proposed a solution using deterministic rules. The other errors are mostly high frequency words that have odd properties, which is an inherently difficult problem most POS taggers are attempting to solve.

Another challenge for English POS tagging is informal or out-of-domain texts, which is the focus of the semi-supervised approach section. With the boom of deep learning in NLP, novel neural networks and pre-trained language models can be expended to unsupervised or semi-supervised learning to further ameliorate the efficiency and tagging accuracy for user generated texts.

Noticeably, as a language pre-processing technique, POS tagging is ultimately intended for improving the performance of complex downstream tasks. Therefore, when pushing for higher accuracy for POS taggers, it is better to prioritize how we can eliminate the errors that have a strong influence on the downstream tasks. In the future, we hope to see POS taggers that are not only accurate and reliable, but also serve their purpose of enhancing more complicated NLP tasks.

5 Text chunking

Text chunking is a NLP task that splits sentences into non-overlapping segments, such as Noun Phrase (NP) and Verb Phrase (VP). Chunking, also called shallow phrasing, can be applied as a pre-processing step before complete parsing. It helps the machine to learn the sentence structure and relation between words, e.g., recognizing names and syntactic components. Thus, it provides a useful foundation for downstream NLP tasks that require a general understanding of sentence components, e.g., NER (Collobert et al. 2011; Yang et al. 2017), text summarization (Gupta et al. 2016), and sentiment analysis (Syed et al. 2014).

In this section, we first review different tagging schemas in text chunking, then investigating previous methods in tow categories: feature engineering approaches and deep learning approaches.

5.1 Tagging schemas

IOB tagging schema Ramshaw and Marcus (1999) innovatively proposed noun phrase chunking as a machine learning problem. Prior to their work, chunk structure was mostly encoded with brackets between words, which is often met with the problem of unbalanced brackets. To solve this problem, they introduced the IOB1 (also known as IOB) tagging schema to represent chunk structures, where "B" stands for the beginning of a chunk that immediately follows another chunk, "I" means the word is inside a chunk, and "O" stands for outside of any chunk. Thus, chunking is considered as a sequence labeling problem. The dataset they derived from the Penn Treebank is later referred to as baseNP and is used in some very early works. Thereafter, Sang and Buchholz (2000) introduced the widely-used dataset CoNLL-2000, extending the task of chunking from noun phrase to other types of chunks, such as verb phrases, prepositional phrases and adverb phrases. The dataset modifies the above-mentioned IOB1 encoding schema to IOB2 (also known as BIO), where "B" is simply used in the beginning of every chunk. It also contains the corresponding POS tag of every token assigned by a standard POS tagger from Brill and Wu (1998). The CoNLL-2000 dataset, along with F-score as metric, has become a standard for evaluating chunkers. Although the CoNLL-2000 comes with IOB2 encoding, it is not difficult to convert it into other schemas. A variety of encoding schemas are explored to study their effects on chunking performance.

IOE tagging schema An alternative to IOB is IOE (Sang and Veenstra 1999), where "E" represents the final word of a chunk immediately preceding another chunk in IOE1, or the final word on every chunk in IOE2. Sang and Veenstra (1999) split the baseNP dataset into two group and investigated the effectiveness of IOB and IOE. Results are inconclusive to determine which one can best improve the performance. Considering that IOB and IOE follow the same core concept to segment chunks, it is reasonable that they do not vary much in performance.

BIOES tagging schema Another popular encoding schema is BIOES (BILOU) (Ratinov and Roth 2009), where "E" stands for the ending token of a chunk, and "S" denotes a single element. Research shows that chunkers using BIOES outperform those using IOB significantly (Yang et al. 2018; Ratinov and Roth 2009; Dai et al. 2015). This is likely because BIOES is more fine-grain then IOB, allowing the machine to learn a more expressive model with only a small amount of extra parameters.

5.2 Feature engineering approach

The feature engineering approaches rely on hand-crafted feature sets from the surrounding contexts, e.g., local lexical information, POS tags, and chunk tags of previous words. In this paper, we further categorize the feature engineering approach into two groups: local classification approaches and global classification approaches.

The local classification approaches view the chunking task as a sequence of classification problems, one for each of the word in the sequence, where the predicted tag at each position may depend on the features of the whole input sentence and the predicted tags of previous words. The global classification approaches, on the other hand, are able to trade off decisions at different positions to obtain a globally optimal label sequence. Chunkers in this category are mostly graphical models, such as HMM (Freitag and McCallum 2000) and CRF.

5.2.1 Local classification approach

The local classification approaches predict the label of one word in a sequence at a time, utilizing different lexical and syntactic information as features, e.g., the word itself, its POS tag, its surrounding words and their POS tags, to make the best local decision.

Ramshaw and Marcus (1999) proposed a chunker with transformation-based learning. The chunk structure is represented by IOB1 tag schema in non-recursive base NP distinction, and by BN, N, BV, V, P in noun/verb phrase separation. First, a baseline heuristic is learned using POS tags. It is then used to produce initial hypotheses for each site in training corpus. When the baseline prediction is incorrect, the rule templates generate candidate rules for different locations based on the identities of words within a neighborhood, their POS tags and the current chunk tags. The candidate rules are tested against the rest of the corpus and sorted based on their positive scores. This will eventually create an ordered sequence of rules that predict the features of words. In order to speed up the learning process, an index is constructed to link each candidate rule to its static locations in the corpus, and the rules are disabled and re-enabled based on their scores and changes. Ngai and Florian (2001) also applied transformation-based learning to chunking, whose method is previously introduced in the POS tagging section.

Daelemans et al. (1999a) proposed a memory-based learning method where POS tagging, chunking, and identification of syntactic relations are formulated as memory-based modules. The proposed model is a lazy learner, keeping all training data available for extrapolation. Thus, it is more accurate than greedy learners for NLP tasks. Memory-based learning constructs a classifier for a task by storing a set of examples. Each example associates a feature vector with one of a finite number of classes. The classifier extrapolates the classes of feature vectors from those of the most similar feature vectors in the memory. The syntactic analysis process is split into a number of classification tasks where input vectors represent a focused item and a dynamically selected surrounding context. Outputs of some memory-based modules are used as input by other memory-based modules.

Based on their work, Sang (2000) proposed a system-internal combination of memory-based learning classifiers to find base chunks. The main idea is to generate five different chunking models by using different chunking representations, namely IOB1, IOB2, IOE1, IOE2, and the bracket structures. Each classifier uses the memory-based learning algorithm IB1-IG (Daelemans et al. 1999a) for determining the most probable tag for each word. The training data is stored and a new item is classified by the most frequent classification among training items closest to the new item. Outputs of the five classifiers are combined using either voting methods, classifier stacking method or combination method. The paper also explored three processing strategies: single-pass, double-pass, and n-pass, where the data are processed once, twice or many passes to identify the correct chunk tags. According to the experiment, the best combination is the Majority voting method and the double-pass method.

Similarly, Van Halteren (2000) proposed a chunking method using Weighted Probability Distribution Voting (WPDV) model (van Halteren 2000). The proposed model has a three-stage architecture. In the first stage, five different base chunkers are trained, including a stacked TiMBL model (Sang 2000), a WPDV model, a reverse WPDV model, a R &M WPDV model, and a LOB WPDV model. Subsequently, another WPDV model is used to combine the outputs of the five base chunkers. Lastly, corrective measures are applied to the systematic errors, which are mostly due to the determination of the start position of NPs.

Koeling (2000) introduced a MaxEnt model for chunking, which is an exponential model that chooses the probability distribution with the highest entropy. MaxEnt operates on the intuition that if there is no evidence to favor one solution over the other, then both solutions are equally likely. Therefore, the probability distribution with the highest entropy should be chosen. Given a word, the probability of a candidate label is dependent on its history, containing the word itself and its tag context. The label with the highest probability is selected.

Kudo and Matsumoto (2000, 2001) employed SVM for text chunking. The paper applied a weighted voting method to 8 pairwise SVM classifiers, using contextual information as features for training. Training data is split into 4 different types of representations, namely IOB1, IOB2, IOE1, IOE2. For each representation, forward parsing and backward parsing are employed, thus creating a weight voting of 8 SVM systems. Four methods are proposed to determine the weight given to each system: uniform weights, cross validation, VC-bound, and Leave-One-Out.

Zhang et al. (2001, 2002) applied a Winnow algorithm (Littlestone 1988; Grove and Roth 2001) to text chunking. Winnow is suitable for problems with a high dimensional feature space. The Winnow multiplicative update algorithm (Littlestone 1988) updates the trainable parameter repeatedly when the algorithm cannot correctly classify an example. However, it may not converge when the data is not linearly separable. The paper proposes regularized Winnow, which converts the original Winnow into a numerical optimization problem that converges in both linear separable case and linear non-separable case, thus making it suitable for NLP tasks such as text chunking. Besides contextual information, the model also uses English Slot Grammar (McCord 1990) as addition features. Dynamic programming (Punyakanok and Roth 2000) is used to determine the best sequence of chunk tags.

Based on their work, Lee and Wu (2007) proposed a mask method based on SVM classifier that does not depend on external knowledge and multiple learners. The purpose of the mask method is to collect unknown word examples from original training data, so that the chunker can handle unknown words in testing data. First, a tokenizer and a POS-tagger is applied to produce POS tag for each token. Then the feature selection component encodes the important features of context words. One-Against-All SVM classification method is employed to classify the IOB1 tag of the words. After lexicon-related features are derived, the training set is divided into 2 or more parts. New training examples can be generated by mapping the new feature dictionary set from each training part. This method emulates training examples that do not contain lexical information, which helps the model considers the effect of unknown words, and adjusts the weights on lexical related features.

Johnson and Zhang (2005) proposed a semi-supervised method for text chunking, which is based on the idea that good classifiers should have similar predictive structure, and thus learns good structure from an auxiliary classification problem can help improve performance on the target problem. The paper presents a linear prediction model for structural learning. Supposedly, there is a low-dimensional predictive structure shared by multiple prediction problems, which can be discovered through joint empirical risk minimization (ERM). The goal of this model is to discover the common low-dimensional predictive structure parameterized by the projection matrix in the predictor, i.e., to find the optimal projection matrix that minimizes the empirical risk summed over all the problems. This optimization problem is solved by alternating structure optimization (ASO) (Ando and Zhang 2005). For semi-supervised learning, the auxiliary prediction problems are generated automatically from unlabeled data. A classifier is trained with a feature map and labeled data, whose behavior is then predicted on the unlabeled data using another distinct feature map. After the training data for each auxiliary problem is created, the optimal projection matrix is computed from the training data via ASO, and the empirical risk on the labeled data is minimized.

5.2.2 Global classification approach

The objective of chunkers using the local classification approaches is to minimize functions related to labeling errors, where they make the best local decisions. As a result, they cannot trade off decisions at different positions against each other to obtain a globally optimal labeling. To solve this problem, many works attempt to establish global classifiers for text chunking. The earliest global classification approach relies on generative graphical models, such as HMM (Kupiec 1992).

Zhou and Su (2000) proposed an error-driven HMM-based text chunking tagger with context-dependent lexicon. The input token sequence is the product of the word sequence and the POS tag sequence. The chunking structure is represented by structural tags, which consists of structural relation, phrase category, and POS tags. The model uses the Viterbi algorithm to find a stochastic optimal tag sequence for the given token sequence. The baseline system only uses the current POS as lexical entry to determine the current structural chunk tag. Then, the paper attempts to add more contextual information by adding lexical entries into the lexicon, such as the current and the previous words, and their POS tags. Adding more contextual information significantly improves the accuracy, however, it is difficult to merge all the above context-dependent lexicons in a single lexicon due to memory limitation. Thus, an error-driven learning approach is adopted to examine the effectiveness of lexical entries and reduce the size of lexicon.

Molina and Pla (2002) proposed an HMM-based tagging method where the model finds the sequence of states of maximum probability given the input sequence. This method can be used in many different shallow parsing tasks including text chunking, given the appropriate input information. When implemented for chunking, the model considers words and POS tags as the input. In addition, the paper suggests that the output tag set could be too generic to produce accurate models. Thus, for a chunking task, the model can be enriched by adding POS information and certain selected words into the chunk tags. These are achieved by applying a specialization function on the original training set. From this new training set, the Specialized HMM can be learned by maximum likelihood. The tagging process is carried out using the Viterbi algorithm.

Although the HMM-based algorithms are well-understood, they require strict conditional independence assumptions to work effectively, which makes it difficult to represent non-independent features, such as surrounding words. Attempts have been made to enable chunkers to handle more statistically correlated features of input tokens while obtaining global optimal labeling, e.g., the perceptron algorithm (Collins 2002) and bidirectional inference algorithm (Tsuruoka and Tsujii 2005), both previously discussed in the POS tagging section.

Another popular algorithm to address this problem is based on CRF, which is the most widely-used alternative to generative graphical models. As mentioned in the POS tagging section, first proposed in Lafferty et al. (2001), CRF is an undirected linear-chain graphical model. It uses a single exponential model for the joint probability of the entire tag sequence given an observation sequence. CRF can not only take in many statistically correlated features from the input data and train them discriminatively, but also trade off decisions at different sequence positions to obtain a globally optimal labeling. Hence, it averts the limitations while maintaining the advantages of the local classification approach and the HMM-based approach.

Sha and Pereira (2003) introduced the application of CRF to text chunking, proposing a novel CRF training algorithm with better convergence properties. For chunking task, the CRF labels are pairs of consecutive chunk tags, which establishes a second-order Markov dependency between chunk tags. The local feature is based on a predicate on the input sequence and current position, and a predicate on pairs of label. Instead of using iterative scaling as training algorithm (Lafferty et al. 2001), the paper experiments with two training methods to maximize the log-likelihood of the training set: preconditioned conjugate gradient (Shewchuk et al. 1994), and limited-memory quasi-Newton (Nocedal and Wright 2006). Both utilize approximate second-order information to achieve high convergence speed.

Following their work, many alternative CRF or other second-order random fields algorithms are proposed for chunkers to model more complex dependencies. For instance, as described in the POS tagging section, Suzuki and Isozaki (2008) employed a SSL CRF, which is also tested to be effective at text chunking.

Sutton et al. (2007) proposed dynamic CRF (DCRF), which is a generalization of the original CRF that repeats structure and parameters over a sequence of state vectors. Compared to conventional CRF, DCRF is able to represent distributed hidden states and complex interactions among states, such as factorial, second-order and hierarchical structure. To achieve this, DCRF introduces clique index c, which represents any state in the unrolled graph through a time step offset and its index in the state sequence y. Then the set of variables in the unrolled version of clique index c at time step t can be denoted as $y_{t,c}$. Let C be a set of clique indices. Similar to standard CRF, given an input sequence x, the conditional probability $P(y \mid x)$ is computed as

$$P(y \mid x) = \frac{1}{Z(x)} \prod_{t} \prod_{c \in C} \exp(\lambda \cdot \mathbf{F}(y_{t,c}, x, t)),$$

where Z is the normalization factor, F is a set of feature function, and λ denotes weights for each feature function. The generalization of DCRF allows for complicated structure, such as the proposed factorial CRF (FCRF), which incorporates edges between co-temporal labels to explicitly model dependencies between different chains. Assume a FCRF has L chains, where $y_{l,t}$ is the variable in chain l at time t. The distribution over output sequence is computed within-chain and between-chain:

$$\begin{aligned} P(y \mid x) = & \frac{1}{Z(x)} \left(\prod_{t=1}^{T-1} \prod_{l=1}^{L} \exp(\lambda \cdot \mathbf{F}(y_{l,t}, y_{l,t+1}, x, t)) \right) \\ & \left(\prod_{t=1}^{T} \prod_{l=1}^{L-1} \exp(\lambda \cdot \mathbf{F}(y_{l,t}, y_{l+1,t}, x, t)) \right). \end{aligned}$$

This factorized structure can be used to jointly train several sequence labeling tasks, such as POS tagging and text chunking, with shared information. Based on this, the paper further describes a marginal DCRF for joint learning between POS tagging and chunking, which is inspired by the notion that the main purpose of POS tagging is to help the prediction of chunking. Therefore, training by maximizing the joint likelihood is not ideal, since the model might trade off accuracy among the chunk tag to obtain accuracy among the POS tag. The proposed marginal training encourages the model to prioritize learning the main task whilst retaining useful information from the other task. That is, in the training set, the observations of POS tag sequence are ignored, thus the model is able to focus on the conditional probability over y. Experiments show that joint training using marginal

FCRF improves chunking accuracy slightly in comparison to cascaded training where the two tasks are learnt in sequence.

Sun et al. (2008) proposed a chunker based on Latent-Dynamic Conditional Random Fields (LDCRF) (Morency et al. 2007) and a coinciding inference algorithm named Best Lable Path (BLP), which can learn latent-dynamics explicitly. More specifically, latent-dynamics is the underlying structure of syntactic contexts, which is often too complex for chunk labels to encapsulate. For example, it is difficult for the BIO tagging schema to differentiate the latent-structures between the sequences "He is her -" and "He gave her-", where the former is likely to be followed by a I-NP tag and the latter a B-NP tag. LDCRF is able to mitigate this problem by explicitly modeling hidden state variables. Given an input sentence $x = (x_1, \dots, x_n)$, the task is to learn the mapping between x and a sequence of labels $y = (y_1, \dots, y_n)$. The model also assumes a vector of hidden state variables $h = (h_1, h_2, \dots, h_n)$ for the sequence. To make training and inference more efficient, the model is restricted to have a disjointed set of hidden states $\mathbf{H}_{y_i}$ associated with each class label y_i. The conditional probability can be written as

$$P(y \mid x, \Theta) = \sum_{h \in \mathbf{H}_{y_1} \times \cdots \times \mathbf{H}_{y_n}} P(h \mid x, \Theta),$$

where $P(h \mid x, \Theta)$ is calculated as conventional CRF:

$$P(h \mid x) = \frac{1}{Z(x)} \exp(\Theta \cdot \boldsymbol{F}(h, x)),$$

where Θ denotes the model parameters. Due to the inclusion of hidden states, the best label sequence $\hat{y}$ cannot be found directly via Viterbi algorithm. Therefore, NLP inference is introduced to search for $\hat{y}$, where the top-k hidden paths over hidden states are chosen using $A*$ search (Hart et al. 1968), and the corresponding probabilities of hidden paths are produced. Subsequently, the estimated probabilities of various label paths can be computed as the sum of the probabilities of hidden paths.

Muis and Lu (2016) proposed a weak semi-Markov CRF model for NP chunking on user generated text, namely NUS SMS Corpus (Chen and Kan 2012). Semi-Markov CRF (semi-CRF) (Sarawagi and Cohen 2004) can be defined as conventional CRF with additional edges from one node to all the nodes up to L words away, representing a segment within which the words will be labeled with a single label. Based on Semi-CRF, the paper introduces a weaker variant that restricts each node to connect either to only the nodes of the same label up to L words away, or to all the nodes only in the next word. With this restriction, weak semi-CRF can decide the next segment length and type separately, whereas Semi-CRF is encouraged to make the decisions simultaneously. This restriction is achieved by splitting every original node into a Begin node and an End node. The End node connects only to the every next Begin nodes of any label. On the other hand, the Begin node connects only to the End nodes with the same label up to L next words. Thus, given the input sequence $x = (x_1, \dots, x_n)$, the conditional probability of label sequence $y = (y_1, \dots, y_n)$ can be defined as

$$P(y \mid x) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^{n}\left(\lambda f(y_{i-1}, y_i, x, i) + \sum_{k=1}^{L} \lambda g(y_i, x, i-k, i)\right)\right),$$

where $Z(x)$ is the normalization factor, λ is function weights, f is the feature function as in standard CRF, and $g(y_i, x, i-k, i)$ represents the feature vector on the edge between the Begin node with the current state y_i at position $i-k$ and the End node with state y_i at position i.

Motivated by the idea that the choice of encoding schema can affect the performance of sequence labeling models, Lin et al. (2020) proposed two Latent Variable CRFs (LVCRF) that can automatically choose the best encoding schema for a given input sentence. They tested the models on the most popular schema, IOB2 and BIOES. The latent variable CRF is achieved by inserting a set of variables between the input and the output based on the chain rule of probability. This allows the model to capture the latent structure between observations and labels. The first model LVCRF-I labels the input sentence by labeling it with two encoding schemas simultaneously and optimizing the parameters to maximize the probability of both schemas. In the training phase, the model is not informed on which encoding schema is better, but trained to maximize the probability of both schemas. The best encoding schema is determined implicitly during the decoding stage when the best labeling path is found via Viterbi algorithm. The second model LVCRF-II, on the other hand, chooses the encoding schema on a word-level, combining the labeling path in two encoding schemas. Therefore, LVCRF-II allows the transformation between the two encoding schemas. Experiments show that LVCRF-II indeed yields higher accuracy. It is also proven that the LVCRF framework can be used in neural network based chunkers such as LSTM to achieve even better result.

5.3 Deep learning approach

Many deep neural networks, e.g., CNN, RNN and LSTM, can be applied to text chunking. Unlike the previous methods, deep learning approaches are able to automatically extract features from the input texts, making it possible to avoid hand-crafted feature templates.

As previously mentioned in the POS tagging section, Collobert et al. (2011) described a window approach for sequence labeling problems. Aside from the proposed model, they also suggested a novel training algorithm for tasks such as text chunking, where tags are organized in chunks and some tags cannot follow other tags. Named sentence-level log-likelihood, it is proposed as an alternative to softmax and CRF, thus allowing the consideration of scores over all possible tag paths for a given sentence. To achieve this, a transition score $A_{i,j}$ is introduced, which is a trainable variable for jumping from tag i to tag j in successive words. The score of a tag path is computed by summing the transition scores and the scores outputted by the neural network. The score over all possible tag paths is normalized via softmax and interpreted as a conditional tag path probability. The advantage of this method over CRF is that it uses a non-linear neural network instead of a linear model to maximize the likelihood, which encourages the model to learn useful features according to the task of interest. Additionally, they jointly train POS tagging, text chunking, and NER using the proposed window approach, where all models share the same lexicon lookup table and the parameters of the first linear layer, and the training objective is to minimize the loss averaged across all tasks.

Similarly, Yang et al. (2017) also utilized the correlation between text chunking, POS tagging and NER. They showed that transfer learning from the latter two tasks to chunking yields competitive performance. Their method was introduced in detail in the POS tagging section.

Huang et al. (2015) proposed a bidirectional LSTM model with a CRF layer (Bi-LSTM-CRF). The model efficiently utilizes past and future input features via a Bi-LSTM layer and sentence level tag information via a CRF layer. Following the above-mentioned work (Collobert et al. 2011), the CRF layer incorporates a state transition matrix as parameters, which enables the model to utilize past and future tags to predict the current tag.

Yang et al. (2016) described a deep hierarchical RNN, which can encode both character level and word level sequential information. Many previous works (Dos Santos and Zadrozny 2014b; Santos and Guimaraes 2015; Kuru et al. 2016) show that character level features help alleviate the out-of-vocabulary (OOV) problem in sequence labeling tasks, most of which rely on convolutional layer to extract such features. The proposed model, however, employs bidirectional GRU (Cho et al. 2014) to achieve this. The model stack multiple recurrent layers together to build a deep GRU. Such deep GRU is used on both character level and word level, together forming a hierarchical GRU. The word representations produced by the hierarchical GRU are fed into another deep bidirectional GRU to extract the context information in the word sequence. The resulting sequence of hidden states is used as input features for the next layer, where a CRF models the dependencies between tags in the sequence and predicts a sequence of tags.

Zhai et al. (2017) proposed a Bi-LSTM-based sequence chunking model where each chunk is treated as a complete unit for labeling. They also explored the idea of using pointer networks (Vinyals et al. 2015) instead of IOB labels. The paper divides sequence labeling into two sub-tasks: segmentation and labeling. The former is to identify scope of the chunks explicitly, whereas the latter is to label each chunk as a single unit based on the segmentation results. The model employs an encoder-decoder framework where the decoder is modified to take chunks as inputs. The Bi-LSTM encoder is used to create a sentence representation as well as segment the input sequence into chunks. It uses a CNNMax layer to extract important information from words in the chunk, and utilizes context word embeddings of the chunks to capture context information. The decoder is a LSTM that takes all the information above to generate hidden states to label the segmented chunks. To further improve the accuracy, the model uses pointer network instead of IOB2 tags to identify chunks. It identifies a chunk, labels it, and repeats the process until all words are processed. At the beginning of a possible chunk, the pointer network determines which word is the ending point. After a chunk is identified and labeled, it serves as the input of the decoder in the next time step. With this setup, the model is able to utilize chunk-level features for segmentation. Experiments show that pointer network yields better performance than the IOB2 encoding schema.

Rei (2017) proposed a semi-supervised multitask learning framework for sequence labeling tasks. They used Bi-LSTM-CRF as baseline model, integrating unsupervised language modeling as the supplementary task, whose objective is to predict the next word in the sequence based on only the hidden state produced by the forward LSTM, and the previous word based on the hidden state from the backward LSTM. This additional language model objective encourages the model to learn more general patterns of semantic and syntactic composition.

Sun et al. (2020) proposed a hybrid neural CRF for multi-view sequence labeling, termed MVCRF, and adopted diverse neural networks for feature extraction of multiple views. The model not only considers the correlation between neighborhood labels and jointly decoding the best label sequence, but also combines multi-view learning by utilizing consensus and complementary principles. The model takes the word view and the POS view of the sequential data as input. Then, it uses Bi-LSTM to extract features from the word view x_1 with pre-trained SENNA word embedding, and uses a linear network for the POS view x_2. The model then regularizes the log-likelihood by the consistency among distinct views to minimize the Euclidean distance between the features across two views in a joint representation space. Then, features from both views are fed into a MVCRF, where the output y is determined by the conditional probability $P(y \mid x1, x2)$.

Liu et al. (2020) proposed a semi-CRF chunker based on stacked Bi-LSTM. Firstly, the word embedding and character encoding vector are concatenated and fed into the lower Bi-LSTM to obtain the sub-word representation. Subsequently, the parameters of the lower Bi-LSTM are extracted and loaded into the upper Bi-LSTM, which takes in the sub-word representation, and outputs the word representation $w = (w_1, w_2, \dots, w_n)$. Based on the existing semi-CRF (Sarawagi and Cohen 2004), the paper describes a new semi-CRF (NSCRF). Let $c = (c_1, c_2, \dots, c_r)$ be a label path of the given sentence, where $c_i = (b_i, e_i, l_i)$ denotes the i-th chunk, b_i represents the beginning word index, e_i the end word index, and l_i the chunk-level tag. Then the conditional probability of NSCRF can be expressed as

$$p(c \mid w) = \frac{1}{Z(c)} \prod_{i=1}^{r} \exp \boldsymbol{F}(l_{i-1}, l_i, w, b_i, e_i),$$

where $Z(c)$ is the normalization factor, $\boldsymbol{F}$ is the feature function.

Wei et al. (2021) proposed a Position-aware Self Attention (PSA) mechanism to address the RNN-based method's limitation of capturing discrete relations in a sentence. The core of this model is the Bi-LSTM-based context encoder, which employs self-attention to encode relative positional information. Specifically, the context encoder contains two self-attentional context fusion layers: one for assigning weights to the initial inputs, the other for re-weighting the output of the Bi-LSTM. The self-attentional context fusion layer learns context-aware representations using PSA mechanism, where the alignment scores between tokens are calculated by a feed-forward neural network with an additional positional bias function consisting of three different positional factors, namely, self-disabled mask bias, distance-aware Gaussian bias, and token-specific position bias. With PSA, the proposed model is able to learn the non-continuous relations between the tokens.

5.4 Summary

In conclusion, existing text chunking algorithms can be categorized as feature engineering and deep learning. Notably, feature engineering methods can be further divided into local and global approaches. The latter is able to leverage global information whereas the former can only make the locally optimal decision. The local approach includes machine learning methods such as transformation-based learning (Ramshaw and Marcus 1999; Ngai and Florian 2001), memory-based learning (Daelemans et al. 1999a; Sang and Buchholz 2000; Sang 2000; Van Halteren 2000), MaxEnt (Koeling 2000), SVM (Kudo and Matsumoto 2000, 2001; Lee and Wu 2007), and window algorithm (Zhang et al. 2001, 2002). For global approach, early works utilize HMM. A significant limitation of HMM is that its strict independence assumptions makes it inefficient to incorporate contextual information. To mitigate this problem, Zhou and Su (2000) took an error-driven approach, whereas Molina and Pla (2002) transformed the training set through specialization functions. CRF (Lafferty et al. 2001; Sha and Pereira 2003), as previously mentioned in the POS tagging section, is more flexible than HMM in terms of feature design. However, its linear-chain structure limits the ability to capture dependencies between non-adjacent states. Some CRF variants are proposed to address this limitation. DCRF (Sutton et al. 2007) is a skip-chain graphical model that enables complex interactions between labels. Similarly, semi-Markov CRF (Muis and Lu 2016) is a skip-chain CRF targeting informal text, which accommodates restricted dependencies between nodes. LDCRF (Sun et al. 2008) captures latent dynamics by establishing a set of hidden states for every class label.

LVCRF (Lin et al. 2020) is able to automatically choose the optimal encoding schema by inserting variables between states and observations.

Compared to feature engineering approaches, deep learning approaches are able to automatically extract features, and better facilitate joint training with other sequence labeling tasks. The most commonly used architecture is RNN variations combined with CRF. Huang et al. (2015) implemented the standard Bi-LSTM-CRF structure. Yang et al. (2016) utilized hierarchical GRU with CRF for better feature extraction. Sun et al. (2020) replaced the standard CRF with MVCRF, where the POS sequence is also incorporated as input. Similarly, Liu et al. (2020) utilized semi-CRF, which adds helpful labeling restrictions to standard CRF. Wei et al. (2021) applied a self-attention mechanism to the Bi-LSTM-CRF structure to model position-aware dependencies. In short, the RNN-CRF structure consistently yields good performance, and the neural network component leaves room for exploration and development. However, such structure is computationally costly, because of the limitation of CRF. Zhai et al. (2017) reframed the chunking task as segmentation and labeling tasks, where a pointer-network is used to identify chunk span, averting the use of CRF. The weakness of this method is that there is limited interaction between the two subtasks to minimize error propagation. Joint learning is another direction of interest. Collobert et al. (2011) and Yang et al. (2017) jointly learned text chunking with other sequence labeling tasks. The former proposed a neural alternative to CRF, which is more adept to learn task-specific features in joint learning. The latter presented a transfer learning framework. Rei (2017) incorporated unsupervised language modeling as an auxiliary task to text chunking.

Table 10 shows a comparison of the features used by the introduced chunkers. Given a focal word, the commonly used features are its POS tag, lexical context (*n* words to its left and to its right), syntactic context (*n* POS tags to its left and to its right). Some chunkers are able to utilize second-order features, e.g., concatenation of the focal word and at least two preceding or following POS tags, or the current POS tag and at least two preceding chunk tags. Such features provide more context but can lead to sparsity problems, and thus are not widely adopted. A few chunkers attempt to capture the structure of a chunk and use chunk-level features, such as the distance from the verb to the chunk head (Daelemans et al. 1999a), or employing a pointer network to learn chunk segmentation before labeling each word (Zhai et al. 2017). Morphological features e.g., affix and capitalization are less prevalent in text chunking, as most of the information they contain are expressed with POS tags. Nonetheless, morphological information can still be helpful in the case of OOV words. Deep learning methods adopt character-level embedding to acquire similar effects. Furthermore, since text chunking is a relatively coarse-grained syntactic task that can benefit from other tasks e.g., POS tagging, NER, or language model, we specially mark out the algorithms that employ joint learning. A summary of the performance of the introduced algorithms are listed in Table 11. The current state-of-the-art model (Akbik et al. 2018) obtained the F1 score of 96.72%.

Compared to other sequence labeling tasks in NLP such as POS tagging and NER, text chunking has received relatively less attention. This has become more apparent in recent years, since most neural network models meant for sequence labeling are largely universal and can be extended to chunking. It is worth exploring whether a model designed specifically for chunking can improve the performance. The more significant challenge, however, is that text chunking as a sub-system in complex applications remains quite rare. Although pushing for accuracy is important, it is also good to keep in mind how chunking as a syntactic pre-processing step can benefit the higher level NLP tasks such as semantics or pragmatics.

Table 10 The comparison between different text chunking methods

Method		POS tag	LC	SC	PC	2nd order	CL	Morph	Character	Joint
FE-local	Ramshaw and Marcus (1999)	✓	✓	✓						
	Ngai and Florian (2001)	✓	✓	✓	✓		✓			
	Daelemans et al. (1999a)	✓	✓	✓	✓					
	Sang (2000)	✓	✓	✓	✓					
	Van Halteren (2000)	✓	✓	✓	✓					
	Koeling (2000)	✓	✓	✓	✓	✓				
	Kudo and Matsumoto (2001)	✓	✓	✓	✓					
	Zhang et al. (2002)	✓	✓	✓	✓	✓				
	Lee and Wu (2007)	✓	✓	✓	✓			✓		
	Johnson and Zhang (2005)	✓	✓	✓	✓			✓		
FE-global	Zhou and Su (2000)	✓	✓	✓						
	Molina and Pla (2002)	✓	✓	✓						
	Sha and Pereira (2003)	✓	✓	✓	✓	✓				
	McDonald et al. (2005)	✓	✓		✓					
	Sutton et al. (2007)	✓	✓					✓		✓
	Sun et al. (2008)	✓	✓	✓		✓				
	Muis and Lu (2016)	✓	✓		✓					
	Lin et al. (2020)	✓	✓	✓		✓				

Table 10 (continued)

Method		POS tag	LC	SC	PC	2nd order	CL	Morph	Character	Joint
Deep learning	Collobert et al. (2011)									✓
	Yang et al. (2017)	✓								✓
	Huang et al. (2015)	✓	✓	✓				✓		
	Yang et al. (2016)								✓	✓
	Zhai et al. (2017)						✓			
	Rei (2017)	✓								✓
	Sun et al. (2020)	✓								✓
	Liu et al. (2020)	✓							✓	
	Wei et al. (2021)								✓	

FE stands for feature engineering. LC stands for lexical context. SC stands for syntactic context. PC stands for preceding chunk tag prediction. CL stands for chunk-level. Morph is short for morphological. Joint stands for joint learning with other tasks

Table 11 The performance of the introduced text chunking methods

Method		Experiment		
		D	R (%)	M
FE-local	Muis and Lu (2016)	NUS SMS	76.62	F1
	Ramshaw and Marcus (1999)	baseNP	92.3	F1
	Daelemans et al. (1999a)	baseNP	93.8	F1
	Ngai and Florian (2001)	CoNLL2000	92.30	F1
	Sang (2000)	CoNLL2000	92.50	F1
	Van Halteren (2000)	CoNLL2000	93.32	F1
	Koeling (2000)	CoNLL2000	91.97	F1
	Kudo and Matsumoto (2001)	CoNLL2000	93.95	F1
	Zhang et al. (2002)	CoNLL2000	93.56	F1
	Lee and Wu (2007)	CoNLL2000	94.22	F1
	Johnson and Zhang (2005)	CoNLL2000	94.39	F1
FE-global	Zhou and Su (2000)	CoNLL2000	93.68	F1
	Molina and Pla (2002)	CoNLL2000	92.23	F1
	Collins (2002)	CoNLL2000	93.53	F1
	Tsuruoka and Tsujii (2005)	CoNLL2000	93.70	F1
	Sha and Pereira (2003)	CoNLL2000	94.38	F1
	Suzuki and Isozaki (2008)	CoNLL2000	95.15	F1
	McDonald et al. (2005)	CoNLL2000	93.90	F1
	Sutton et al. (2007)	CoNLL2000	93.87	F1
	Sun et al. (2008)	CoNLL2000	94.34	F1
	Lin et al. (2020)	CoNLL2000	92.44	F1
	Muis and Lu (2016)	NUS SMS	76.62	F1
Deep learning	Collobert et al. (2011)	CoNLL2000	94.32	F1
	Yang et al. (2017)	CoNLL2000	95.41	F1
	Huang et al. (2015)	CoNLL2000	94.49	F1
	Yang et al. (2016)	CoNLL2000	95.41	F1
	Zhai et al. (2017)	CoNLL2000	94.72	F1
	Rei (2017)	CoNLL2000	93.88	F1
	Sun et al. (2020)	CoNLL2000	95.44	F1
	Liu et al. (2020)	CoNLL2000	91.80	F1
	Wei et al. (2021)	CoNLL2000	95.15	F1
	Zhao et al. (2019)	CoNLL2000	94.80	F1
	Akbik et al. (2018)	CoNLL2000	96.72	F1

FE stands for feature engineering. D denotes dataset. R denotes result. M denotes measure

With the development of neural network, the performance of sequence labeling model is advancing rapidly. Currently, the most popular approach to text chunking is the combination of different deep learning algorithms and CRF. In the future, one possible way to further improve the performance is to apply graph neural network to chunking models. Another way is to find a neural network alternative to CRF. Lastly, we hope to see more integration between text chunking and other more complex downstream tasks, e.g., aspect extraction, sentiment analysis, etc.

Table 12 Widely used corpora for lemmatization

Dataset	Description	Reference
CoNLL-2007	Dependency parsing corpus	Nivre et al. (2007)
CoNLL-2009	Syntactic and semantic dependencies corpus	Hajič et al. (2009)
CoNLL-2018	Multilingual parsing corpus	Zeman et al. (2018)
UD	The Universal Dependencies treebanks	Nivre et al. (2016)
Multext	Multilingual text tools and corpora	Ide and Véronis (1994)
MULText-EAST	The MULText-EAST Slovene lexicon	Erjavec (1998)
UniMorph	The Universal Morphology project	Kirov et al. (2018)
PDT	The Prague Dependency Treebank	Böhmová et al. (2003)
IFD	The Icelandic Frequency Dictionary	Helgadóttir (2012)
CELEX	Lexical databases of English, Dutch and German	Jongejan and Dalianis (2009)
Cast3LB	Spanish treebank	Civit and Martí (2004)
CESS-ECE	Multilingual and multilevel annotated corpus	Martı et al. (2007)
1984	Manually annotated G. Orwell's *1984*	Gesmundo and Samardzic (2012)
GML	The Middle Low German dataset	Peters and Nagel (2014)
PATB	Penn Arabic Treebank	Maamouri et al. (2004)
ARZ	Egyptian Arabic Morphological Annotation	Maamouri et al. (2012)
FTB	French Treebank	Seddah et al. (2013)
L-Lem	Lexical databases of Modern Greek and English	Lyras et al. (2007)
D-Lem	Afrikaans lexicon	Daelemans et al. (2009)
Lem-list[a]	Lexical databases of 25 languages	Akhmetov et al. (2020)
RELIG	Religious texts in Middle Dutch	Van Kerckvoorde (2019)
CG-LIT	Literary texts in Middel Dutch	Van Kerckvoorde (2019)
WSD[b]	Hindi word sense disambiguation health and tourism corpora	Khapra et al. (2010)
LT4HALA[c]	Latin texts	Celano (2020)
TüBa-D/Z	The TüBa-D/Z treebank for German	Telljohann et al. (2004)
NoSta-D	German texts with non-standard variations	Dipper et al. (2013)
UD-EWT	The English Web Treebank	Silveira et al. (2014)
SIGMORPHON	The SIGMORPHON 2019 shared task	McCarthy et al. (2019)
SEJFEK	Polish economic lexicon	Savary et al. (2012)

[a]https://github.com/michmech/lemmatization-lists

[b]http://www.cfilt.iitb.ac.in/wsd/

[c]https://circse.github.io/LT4HALA/

6 Lemmatization

Lemmatization is a NLP task that reduces the inflected forms of given words into their morphologically correct root forms. It is an essential pre-processing technique that extracts concepts and keywords for downstream applications, e.g., search engine (Halácsy and Trón 2006; Balakrishnan and Lloyd-Yemoh 2014) and dialogue systems (Zhao and Gao 2017; Altinok 2018; Liu et al. 2019). Another commonly used method is stemming, which also converts words into their base form, but does so by cutting off the prefixes or suffixes.

Lemmatization, on the other hand, performs a morphological analysis based on the context of given words, and thus is able to preserve more syntactic information. For example, given the word "*studied*", a stemmer simply removes the suffix and returns "*studi*", whereas a lemmatizer is able to extract the proper lemma "*study*".

Lemmatization has received growing attention in recent years, especially for highly inflected languages such as Dutch, Latin and Arabic. As lemmatizing English words are relatively easy, here we cover lemmatizers for inflection-rich languages to provide more perspectives to this task. Annotated training corpora for lemmatization mostly include lexicons for the target languages, CoNLL-2007 (Nivre et al. 2007), CoNLL-2009 (Hajič et al. 2009), CoNLL-2018 (Zeman et al. 2018) and the Universal Dependencies (UD) treebanks (Nivre et al. 2016). The standard evaluation metric is accuracy. Relevant information about the commonly used lemmatization datasets can be viewed in Table 12.

The existing lemmatizers regard lemmatization as either a suffix and prefix transformation problem, or a string-to-string transduction problem. The former focus on the starting and ending letters, identifying recurring affixes and transforming them. The latter, on the other hand, considers the whole word form and generates the operations to convert it into its lemma. In this paper, we introduce the previous works in three section: the transformation approaches, the statistical transduction approaches, and the neural transduction approaches.

6.1 Transformation approach

The early lemmatizers learn a set of classification rules that detects and modifies the suffix and/or prefix of a given word form to transform it into the corresponding lemma. The transformation approaches view lemmatization as a rule-based classification problem, where the class label assigned to a word is defined via giving the transformation to be applied on the word in order to get the normalized form. For instance, class labels can take the form of *(x to y)*, where x is the suffix of the word form and y is that of its lemma.

Mladenic (2002) proposed two methods for mapping from words to their lemmas. The first on is letter-based representation using transformation-based learning, where the machine learns a set of classification rules from feature set comprised of suffixes. The other method is context-based representation using Naïve Bayes Majority classifier, where the features are n-grams of the given words. The experiments show that the rule-based approach performs better.

Plisson et al. (2004) proposed a lemmatizer based on Ripple Down Rules (RDR) (Compton et al. 1992) induction algorithm. As opposed to the if-then classification rules, RDR creates exceptions to existing rules, so that the addition of new rules is confined to the context and will not cause inconsistency in the rule base. New RDR rules are added by creating "except" or "else" branches to the existing "if-then" rules, creating a tree-like decision structure. The model is trained on a lexicon of lemmatized Slovene words to learn what suffixes should be removed and/or added to get the normalized form. Results show that the RDR approach achieves better accuracy than the standard classification rules.

Juršič et al. (2007) further explored the application of RDR in the automatic construction of lemmatizers, presenting the LemmaGen system. They improved the original RDR method by instantiating general concepts in lemmatization into domain specific terms, such as organizing the training samples into *(word-form, lemma)* pairs, and restricting the form of rule condition and consequent. The LemmaGen is trained and tested on lexicons from multiple languages. It is proven to consistently outperform the original RDR.

Erjavec and Džeroski (2004) presented a lemmatizer for unknown Slovene words. Their method consists of two steps. First, a trigram tagger (Brants 2000) performs morphosyntactic description tagging on the input text. Subsequently, the resulting morphosyntactic tag and the word itself are passed onto a first-order decision list learning system (Manandhar et al. 1998), which learns ordered sets of classification rules. The lemmatizer is trained on a lexicon containing lemmas with their full inflectional paradigms.

A limitation of the rule-based methods above is that classification rules are learned based on fixed-length suffixes. An alternative is to find the LCS between a word form and its lemma to identify the possible suffix.

Kanis and Müller (2005) constructed a lemmatizer automatically from the Full Form - Lemma (FFL) training dictionary using the LCS approach. They focused on three types of OOV problems, namely, missing full forms, compound words, and unknown words. To address the former two, they introduced the Hierarchical Lemmatization without Rule Permission Check (HLWRPC), where a new lemmatization algorithm without the lemmatization rule permission check is used when encountering missing full forms or compound words. To address the unknown words problem, a set of word applicable rules is assigned to every word. When lemmatizing an unknown word, the word applicable rules create a set of prefix and suffix patterns. Then these patterns are matched in the relevant table. The rules with the highest count of winnings are applied on the unknown word.

Jongejan and Dalianis (2009) described a method that automatically generates classification rules to handle not only suffix but also prefix and infix changes to transform word forms into lemmas. Such affixes are identified via finding the LCS and their positions in relation to the LCS. In the training phase, the system tentatively lemmatizes the full form in each training pair by selecting from the rules that have been created. If the selected rule produces a wrong lemma, a new rule is incorporated into the rule base so that the new rule is selected instead of the erroneous rule and generates the right lemma from the full form. The training procedure terminates when the full forms in all the training pairs are transformed to their corresponding lemmas. Then the system returns a data structure containing a set of classification rules that a lemmatizer must traverse to arrive at one rule that is selected to fire. They explored two different data structures to store rules, namely Directed Acyclic Graph (DAG) and plain tree structure with depth first, left to right traversal. Experiments show that the latter is more efficient.

Daelemans et al. (2009) applied prototype-based Active Learning (AL) to memory-based lemmatization. AL (Cohn et al. 1996) is a type of unsupervised approach where criteria are investigated to allow ordering the unannotated data in a way that the instances potentially contributing most to the speed of learning can be annotated first. The proposed lemmatizer is inspired by a novel AL algorithm called Prototype-Based Active Classification (PBAC) (Cebron and Berthold 2009), where a new labeled prototype is added in each learning iteration to fine-tune the classification. Prototypical examples are selected first, whereas examples at the classification boundary are only selected automatically when it becomes necessary. To apply the PBAC approach to lemmatization, they used word frequency and word length as features, assuming that longer or low frequency words are less prototypical than shorter or high frequency words. They hypothesized that contrary to the standard PBAC, less prototypical linguistic examples should provide better results faster in AL. Therefore, less prototypical instances are added to the memory-based classifier (Daelemans et al. 2004) at the start of the learning process. The classes are automatically generated based on the LSC. Experiments show that their algorithm indeed outperforms random data selection and other AL methods in lemmatization.

Gesmundo and Samardzic (2012) formalized lemmatization as a category tagging task, where a word-to-lemma transformation rule is encoded as a single label. Specifically, given a word form, its prefix and/or suffix are found using the LCS method. Then, a transformation rule is defined by a tuple that contains the four types of generic transformation, namely, removing a suffix, adding a new lemma suffix, removing a prefix, and adding a new lemma prefix. Such transformation rules are used as labels. With this setup, any supervised tagging model can be used as a lemmatizer for any languages.

6.2 Statistical transduction approach

As opposed to transforming word forms into lemmas by modifying suffix and/or prefix, the transduction approach views lemmatization as a character-level word-to-lemma transduction process. The introduction of edit tree paves the foundation for this approach, which converts an input string into the output string. Alternatives include edit distance, word vectors, etc.

Chrupała (2006) was the first to propose a data-driven, context-sensitive lemmatizer based on a class-inference mechanism called the Shortest Edit Script (SES), which is able to detect recurring patterns in the mappings from words to their corresponding lemma, thus automatically deriving the lemmatization classes from training data. Specifically, an edit script of two sequences is a set of instructions that specifies the transformations from one sequence to the other. These instructions take the form of inserting or deleting a character at a designated position. Hence, finding the LCS is integral to finding the SES between two sequences. Though unlike the previously mentioned LCS-based transformation approach, SES does not identify suffix or prefix based on its position to map onto that of the lemma, but generating character-level modifications instead. This approach is proven to perform well for different languages, effectively reducing the labor to manually create full-paradigm inflectional lexicon.

Based on the SES mechanism, Chrupała et al. (2008) and Chrupala (2010) proposed Morfette, which is a modular, data-driven, probabilistic system that jointly learns lemmatization and morphological tagging from morphologically annotated corpora. Morfette consists of two MaxEnt classifiers (Ratnaparkhi 1996) that are trained to predict lemmas and morphological tags respectively, and another module that dynamically combines their predictions and outputs the probability distribution over tag-lemma pair sequences. The lemma classifier uses the SES method to induce classes automatically.

Taking inspiration from the above-mentioned works, Müller et al. (2015) presented LEMMING, a modular model that jointly learns lemmatization and morphological tagging at the token level. The proposed lemmatizer maps an inflected form into its lemma given its morphological attributes using a log-linear model. Following the induction method in Morfette (Chrupała et al. 2008), the lemmatizer selects candidates through a deterministic pre-extraction of edit trees. This formalization allows the integration of arbitrary global features. It is vastly used in later transduction-based models. For joint learning, the lemmatizer is combined with a morphological tagger MARMOT (Müller et al. 2013) in a tree-structured CRF. Experiments show that LEMMING yields significant improvements in joint accuracy compared to Morfette and other lemmatizers.

Lyras et al. (2007) implemented the Levenshtein edit distance on a dictionary-based algorithm for automatic lemmatization. Given an input word, the system calculates the string similarity between the input and all the lemmas stored in a dictionary, producing a set of lemmas with minimum edit distance from the input word. To further improve the algorithm, they created a list containing the most common suffixes used in the target language to initiate the system. In this case, given a word, the system removes the first possible suffix of the input, calculating the edit distance between the stemmed input and lemmas. The lemmas with the minimum distance is stored. This process is repeated until all possible suffixes of the input word are removed. Then the system compares the edit distance of all the stored lemmas, and returns the *n*-best ones with the overall least edit distance. Experiments show that both methods perform well, but the stemming method, although more labor-taxing, indeed achieves higher accuracy than the baseline edit distance algorithm in both modern Greek and English.

Dreyer et al. (2008) presented a conditional log-linear model, which employs overlapping features over latent alignment sequences, and learns latent classes and latent string pair regions from incomplete training data. Given an input, the candidate output is selected by a sliding window over the aligned *(input, output)* pair. At each window position, the log probabilities of all possible alignments are accumulated, evaluating each alignment separately. To further improve the performance, new latent dimensions can be added to the *(input, output)* tuple.

Toutanova and Cherry (2009) presented a global joint model for lemmatization and POS tagging trained on morphological lexicons and unlabeled data. The model consists of two components - a semi-supervised POS tagger (Toutanova and Johnson 2007), and a lemmatizing transducer that is optionally given the POS tags of input word to inform the lemmatization. Taking a pipeline approach, given a sentence, the POS tagger first predicts a set of tags for each word. Subsequently, the lemmatizer predicts one lemma for each of the possible tags. The *k*-best predictions of tag sets and lemmas are chosen to be the output. In addition, based on the notion that if two words have the same lemma, their POS tag sets should be dependent, the dependencies among multiple words are dynamically determined. Experiments show that their joint learning model outperforms the direct transduction lemmatizer in different languages.

Nicolai and Kondrak (2016) presented three lemmatization methods leveraging inflection tables. The first one is a stem-based lemmatizer, which is built upon a word-to-stem transduction model by adding a stem-to-lemma model, both of which are adapted from the DIRECTL+ transducer (Jiampojamarn et al. 2010). The word-to-stem model is trained on morphological segmentations annotated by leveraging paradigmatic regularity in inflection tables, whereas the stem-to-lemma model is trained on character-aligned pairs of stems and lemmas. The second model is a stemma-based lemmatizer, consisting of a word-to-stemma model and a stemma-to-lemma model. The word-to-stemma model is an improved word-to-stem model, where inflection tables are further utilized by replacing stems with the corresponding stemmas, and affixes with inflection tags. In this way, the boundaries between stems and affixes are identified by tags. The stemma-to-lemma model is the same as the stem-to-lemma model except it is trained stemma-lemma pairs. The final method is direct word-to-lemma transduction, which is trained on word-forms paired with their lemmas and inflectional tags obtained from the inflection tables. Furthermore, the paper employs a re-ranking algorithm (Joachims 2002) after the *n*-best lists of possible lemmas are generated, which is able to leverage large, unannotated word lists to improve the accuracy. Experiments show that the re-ranking algorithm indeed boost the performance, and the direct lemmatization method outperforms the other two models in most languages.

Barteld et al. (2016) presented a novel candidate ranking method to tackle two challenges in data-driven lemmatization. First, highly inflected languages can feature complex morphological changes such as prefix and/or infix modification. Secondly, spelling variations can appear in non-standard texts. To bring more emphasis on word-internal modifications, Lexical Correspondence (LC) is used for lemma candidate generation instead of the commonly used edit trees. An LC is defined as a tuple (T, L) where T and L are two sequences of constants and variables with the requirement that the same variables appear in both sequences. Given a word form and an LC, the constants in sequence T are matched with characters in the word form and the variables are replaced with the remaining substrings. Then, the corresponding lemma can be read off the second sequence L. To restrict over-generalization, insertions are anchored by their character offset either from the beginning or the end of the input word form. To deal with spelling variation, given an OOV word form, the lemmatizer generates lemma candidates from similar IV word form. The similarity is measured by he Levenshtein distance.

Gallay and Šimko (2016) innovatively proposed a method that automatically constructs lemmatizer by using text corpora to build vector models of words to infer lemmas. The core concept is that vector space word representation encodes not only syntactic and semantic patterns, but also morphological ones. Given an input word, the relevant reference *(word, lemma)* pairs are chosen from the reference lexicon by applying vector shifts to the input word to retrieve all candidate lemmas. Then, each candidate is assigned a weight based on their similarity with the input word. The correct lemma candidate is expected to appear in connection with different reference pairs. Therefore, the weights for the same candidate are summed together. The candidate with the highest total weight is outputted as the correct lemma.

Rosa and Žabokrtský (2019) presented an unsupervised approach by performing agglomerative clustering of word forms with a novel distance measure. With the assumption that the inflected forms of the same lemma tend to be similar in both spelling and meaning, the proposed distance measure is a combination of string similarity that is given by Jaro-Winkler edit distance (Winkler 1990), and the cosine similarity of FastText word embeddings.

Akhmetov et al. (2020) introduced a language-independent lemmatizer based on the Random Forest classification algorithm. Firstly, all words are converted into vectors by a TF-IDF vectorizer (Luhn 1957; Jones 1972) in the form of a sparse matrix. Then the transpose of this matrix is multiplied by itself to produce a character co-occurrence matrix, every row or column of which serves as character embedding. For every word-lemma pair in the given dictionary, the word form is encoded by the character co-occurrence matrix, whereas the lemma is encoded by the character ordinal numbers. For lemmatization, a Random Forest classifier with bootstrapping is adopted. Experiments show that this simple lemmatizer performs well across a wide range of languages.

6.3 Neural transduction approach

The recent advancement of neural networks brings a fresh perspective to the transduction approach. The encoder-decoder architecture especially receives a lot of attention, as character-level Seq2Seq model is exceptionally adept at handling the string-to-string transduction task, able to capture more contextual information than the statistical approaches.

Kestemont et al. (2017) described a deep learning approach to lemmatization for variation-rich languages. The proposed system consists of two basic components. One is temporal convolutions that model the orthography of input words at the character level. The other is distributional word embeddings from Skip-gram model, which represents the lexical context surrounding the input. Given a word, the system feeds the focus token into the convolutional component, and its left and right tokens into two embedding components, respectively. The outputs of these three sub-nets, along with the one-hot encoding of the input token, are concatenated to form one single hidden representation, which is then passed through a final linear layer to produce the lemma.

Chakrabarty et al. (2017) introduced a composite Bidirectional Gated Recurrent Neural Network (BGRNN) architecture for language-independent, context-sensitive lemmatization. Following previous works (Chrupała et al. 2008; Müller et al. 2015), the task is defined as detecting the correct edit tree representing the transformation between a word form and its lemma. The proposed architecture consists of two stages. First, a BGRNN is used to extract the character level dependencies. The outputs are combined with the corresponding word embedding given by Word2Vec (Mikolov et al. 2013) to form the final word representations. Then, the representations are fed sentence-wise into the second BGRNN to capture the contextual information of the input word, and to learn the mapping from word embeddings to word-lemma transformations. To further improve the performance, a subset of unique edit trees involving irregular transformations are sorted out from the corpus before training. This prior knowledge is represented as applicable tree edit vectors, and combined with the hidden states form the second BGRNN to be passed on to the final classification layer. For the BGRNNs, the paper explores the combination of two BiLSTMs and two bidirectional GRUs. Experiments show that the former achieves higher accuracy.

Bergmanis and Goldwater (2018) proposed a context-sensitive lemmatizer called Lematus, which is based on the standard attentional encoder-decoder architecture, and incorporated character-level sentence context. The system is based on a NMT model (Sennrich et al. 2017), which consists of a 2-layer bidirectional GRU encoder, and a 2-layer decoder with a conditional GRU (Sennrich et al. 2017) in the first layer and a standard GRU in the second layer. The input sequence is a space-separated character representation of a given word in its *n*-character left and right sentence context. With this method, context is modeled using solely the character contexts of the focus word, as oppose to additional POS information or word embeddings trained on large corpus.

To address the problem of low-resource languages, Bergmanis and Goldwater (2019) proposed a semi-supervised method that combines type-based learning and context-sensitive approach. Using the above-mentioned lemmatizer Lematus as a baseline, they incorporated unambiguous word-lemma pairs in the inflection table from the Universal Morphology project (UniMorph) (Kirov et al. 2018) as additional training data. They also collected sentence context for them from Wikipedia. Experiment results indicate that this data augmentation indeed improves a baseline lemmatizer trained on low-resource language corpus.

Celano (2020) introduced a joint lemmatizer and POS tagger for Latin. Given a sentence, gradient boosting machine called LightGBM (Ke et al. 2017) predicts the corresponding POS tag sequence. Subsequently, the POS labels are combined with word embedding pre-trained from large Latin corpora, and fed into a standard Seq2Seq model to output the lemma sequence.

Arakelyan et al. (2018) presented a LSTM-based neural network that jointly learns lemmatization, POS tagging, and morphological feature extraction. Given a sentence, each word is represented by a concatenation of three vectors - a pre-trained word vector produced by FastText, a one-hot representation of casing features, and a character-level representation extracted by a BiLSTM. The concatenated word representation is passed through three layers of LSTM to obtain the hidden states containing information about lemma, POS tags, and morphological features. For POS tagging and feature extraction, a linear layer is applied as the output layer. For lemmatization, a GRU-based decoder is added per word, all of which share weights and work in parallel. Finally, the output of each decoder is fed to another linear layer, which produces a sequence of characters that form the predicted lemma.

Pütz et al. (2018) proposed a morphologically-informed Seq2Seq-based lemmatizer. The model extends the standard encoder-decoder architecture with Luong-style monotonic attention (Luong et al. 2015; Raffel et al. 2017). The hidden state outputted by the encoder is concatenated with the embedded morphological and POS tags, passing through a feed-forward layer with SELU activation function (Klambauer et al. 2017), and fed into the decoder to generate the predicted lemma. The proposed model is compared with LEMMING (Müller et al. 2015) in German, achieving competitive accuracy. The results also indicate that a log-linear lemmatizer such as LEMMING is preferable when dealing with misspelled words, whereas Seq2Seq lemmatizer is able to generalize and handle OOV words better.

Kondratyuk et al. (2018) proposed LemmaTag, which is a featureless bidirectional-RNN-based architecture that jointly learns lemmatization and POS tagging. Given a sentence, a GRU outputs the character-level embedding for every word, which is summed with the word embedding to form the final word embeddings. The resulting sequence is passed onto two layers of BiLSTM with residual connections, producing a sequence of word representations with sentence-level connections. The POS tagger, made up of a fully-connected layer, predicts the tag values, concatenating them into a flat vector to pass onto the lemmatizer. The lemmatizer consists of a LSTM layer with character-level attention mechanism that takes in the the final word embedding, the character embedding, and the POS features of the focus word to generate the corresponding lemma.

Malaviya et al. (2019) introduced a simple LSTM-based joint learning model for lemmatization and morphological tagging. Given a sentence, the morphological tagger obtains the word representation for each word using a character-level BiLSTM, which is then fed into a word-level BiLSTM to predict the corresponding morphological tag. For lemmatization, a string-to-string transduction model (Wu and Cotterell 2019) is adopted, which is a Seq2Seq model with hard attention mechanism (Xu et al. 2015a; Rastogi et al. 2016). The joint probability of the input sentence is define as the product of the probability outputted by the tagger and all the probabilities outputted by the transducer.

Yildiz and Tantuğ (2019) proposed Morpheus, a joint contextual lemmatizer and morphological tagger. Similarly to the above-mentioned works, given a sentence, firstly a character-level LSTM generates word vectors, which are then fed into a word-level Bi-LSTM to produce context-aware word representations. Subsequently, two separate LSTM decoders are employed to predict morphological tags and edit operations from word forms to lemmas. More specifically, to find the minimum edit operations, a dynamic programming method based on Levenshtein distance is used.

Manjavacas et al. (2019) also built upon the classic encoder-decoder architecture to acknowledge the difficulty of spelling variations in lemmatization for non-standard language. Specifically, they presented a hierarchical sentence encoder that is jointly trained for lemmatization and language modeling, adopting the attention mechanism (Bahdanau et al. 2014) to extract additional context. The hierarchical encoder consists of three levels. A bidirectional RNN first computes the character-level representation, which is fed into another bidirectional RNN to extract word-level features. Lastly, the final bidirectional RNN outputs the sentence-level features based on the word-level hidden state. To extract even higher quality sentence-level features, an additional word-level bidirectional language model is incorporated into the lemmatizer. Two softmax classifiers are used to predict the left and right tokens using the forward and backward sentence-level hidden states, respectively. The losses of lemmatization and language modeling are jointly minimized. The proposed lemmatizer is thus able to represent global sentence context information without the need for POS or morphological tags.

Kondratyuk (2019) presented a cross-lingual joint learning model for lemmatization and morphological tagging using multilingual BERT (Devlin et al. 2018). Given an input sentence, a pre-trained multilingual cased-BERT is used to encode the tokens in the sentence. Subsequently, the encoded tokens are passed through two separate layer attention (Kondratyuk and Straka 2019) to produce embeddings that are specific to each task. Furthermore, the model also applies BiLSTM character embeddings (Ling et al. 2015b; Kim et al. 2016) to generate enhanced morphological representations that are summed with the two task-specific embeddings, respectively. Following his previous work (Kondratyuk et al. 2018), the decoder for each task consists of two successive layers of word-level bidirectional residual LSTMs. For lemmatization, the final hidden state is fed into a feed-forward layer to map the focus token to one of the pre-computed SES (Chrupała 2006) that transforms it to the correct lemma. Similarly for morphological tagging, a feed-forward layer takes in the hidden state and predicts the morphosyntactic description (MSD) class.

Taking joint learning a step further, Zalmout and Habash (2019) proposed a Seq2Seq-based model that handles both lexicalized and non-lexicalized features. It jointly learns lemmatization, diacritization, normalization, as well as fine-grained morphological tagging. This is achieved by means of applying parameter sharing strategies on top on the standard LSTM encoder-decoder architecture. On one hand, different non-lexicalized features are modeled on word-level by a Bi-LSTM tagger that shares parameters with the encoder. One the other hand, all lexicalized features are handled on character-level by the same encoder, and then sent to their corresponding decoders separately. The proposed model gains improvement across all of the joint tasks in two Arabic corpora.

Unlike previous contextual neural network approaches, which takes into account the whole sentence, Chakrabarty et al. (2019) hypothesized that a limited context is sufficient for lemmatization. Based on this notion, they introduced a BiLSTM-CNN lemmatizer. Given a sentence, a BiLSTM is applied to extract character-level embeddings, which are then arranged into a matrix and processed by a CNN to generate positional embeddings. The size of filters in the CNN is determined by the length of the context to include, which is set to be the trigram of focus tokens in the paper. Experiments show that BiLSTM-CNN indeed outperforms the BiLSTM-BiLSTM lemmatizer (Chakrabarty et al. 2017).

Schmitt and Constant (2019) focused on the lemmatization of multiword expressions (MWEs), which are combinations of several words that display the linguistic properties of a lexical unit, but are present in the lexicon like simple words. The main challenge of lemmatizing MWE is that its unique properties require different classification rules on top of simple-word lemmatization knowledge. To acknowledge this problem, they propose a deep encoder-decoder lemmatizer solely based on the internal context of MWE. Given a MWE, a GRU encodes the character-level word embeddings, which are passed onto another GRU as a sequence to extract the internal context. The final word representation in an MWE is a concatenation of the character-level word embedding, its left internal context, its POS tag, its position and the length of the MWE. Then, a character-level conditional GRU with an attention mechanism decodes the final word representation to generate the corresponding lemma. Although the model generalizes well on unknown MWEs, the paper also addresses its limitations. First, the input MWE and the produced base form must contain the same number of words. Second, due to the one-to-one correspondence of its architecture, the model has difficulties in modeling lemmatizations that modify the word order.

Zalmout and Habash (2020) presented a character-level Seq2Seq lemmatization model, utilizing several sub-word features, namely FastText *n*-grams, greedy stem, orthographic roots, and orthographic patterns. They used Lematus (Bergmanis and Goldwater 2018) as a baseline. Instead of fixed-size character window, they utilized the characters of the *n* words surrounding the focus word, employing LSTM instead of GRU. They explored two configurations to incorporate sub-word features. At the encoder side, they condition on different sub-words by concatenating their embeddings with the character embeddings before feeding them into the encoder. The embeddings for sub-words can be learned as a part of the system, or pre-trained on an external corpus. Alternatively, greedy stem, roots, and patterns can also be learned jointly as auxiliary tasks at the decoder side. In this configurations, lemmatization and the auxiliary tasks share the same encoder, whilst having specific decoders for each task. Experiments indicate that though all the proposed methods outperform the standard Seq2Seq lemmatizer, incorporating pre-trained FastText *n*-grams at the encoder side yields the best result.

Milintsevich and Sirts (2021) proposed a novel hybrid approach that enhances the Seq2Seq lemmatizer with external resources. Their model is based on the lemmatization module from the Stanford parsing system (Qi et al. 2018, 2020), which takes the character-level representation and the corresponding POS tag into a BiLSTM encoder and a BiLSTM decoder with attention, outputting the predicted lemma using greedy decoding. To improve the performance of this baseline, they incorporated additional lemmas extracted from an external lexicon or a rule-based system. The model employs another encoder, which receives and processes the candidate lemmas from an external lexicon, or generated by a rule-based system. The decoder then learns via two separate attention mechanism to predict lemma using the combination of the classic string-to-string transduction as well as copying lemma characters from external candidates. Results indicate that the enhanced lemmatizer performs considerably better than a simple lexicon extension method based on the Stanford system.

6.4 Summary

In conclusion, we divide the existing lemmatizers into two categories - transformation approaches and transduction approaches. Transformation approaches regard affix as the base unit for rule-based classifications. In early works, the machine can only learn rules that transform suffixes of fixed length (Mladenic 2002; Plisson et al. 2004; Juršič et al. 2007; Erjavec and Džeroski 2004). Such limitation can be solved by using LCS, which enables the machine to identify prefixes and suffixes of any length. Kanis and Müller (2005) specifically targeted missing full forms, unknown words, and compound words. Jongejan and Dalianis (2009) focused on detecting prefix and infix as well as suffix. Daelemans et al. (2009) utilized PBAC to enable efficient unsupervised learning. Gesmundo and Samardzic (2012) reframed lemmatization as a category labeling task, so that any tagging model can be applied as a lemmatizer.

Transduction approaches, on the other hand, view either character as the base unit for lemma generation, or the entire word as the base unit to map onto its lemma based on string similarity measures. Transduction-based methods are more flexible than the transformation-based ones, as they do not rely on an affix lookup table, manually-built or automatically-generated. Thus, transduction approach is better suited to handle irregular changes. In this paper, we further split the transduction approaches into statistical and neural-network-based methods, since the development of deep learning brings a new perspective to lemmatization. Statistical transduction approaches mostly rely on edit scripts, for instance, the data-driven SES mechanism (Chrupała 2006; Chrupała et al. 2008; Chrupala 2010). Müller et al. (2015) improved SES by proposing edit tree, which does not encodes the LCSs and thus more capable of generalizing. Following their work, Lyras et al. (2007); Nicolai and Kondrak (2016) enhanced lemmatization with stemming. Toutanova and Cherry (2009) incorporated POS tagging as a sub-system. Barteld et al. (2016) further improved edit tree with LC, which is more adept at handling word-internal inflections. Dreyer et al. (2008) proposed a window approach as an alternative to edit tree, which can handle more complex inflections, while it is restricted by a window size. Another alternative is to infer lemma from word embeddings (Gallay and Šimko 2016; Rosa and Žabokrtsk\`y 2019; Akhmetov et al. 2020), paving the way for neural approaches.

Neural transduction approaches rely on neural networks to automatically learn the character-level correspondence between lemmas and words. Earlier methods use various types of neural networks to produce character embeddings, concatenate it with word embeddings, and use another neural network to predict the corresponding lemma (Kestemont et al. 2017; Chakrabarty et al. 2017, 2019). Recent works mostly adopt the character-level encoder-decoder architecture (Bergmanis and Goldwater 2018; Pütz et al. 2018). Schmitt and Constant (2019) modified the encoder-decoder structure to specifically target MWEs. Bergmanis and Goldwater (2019) applied SSL to the architecture for low-resource domain. Milintsevich and Sirts (2021) enhanced Seq2Seq lemmatization with external resources. Joint learning is often applied to improve the performance of lemmatization. Commonly used auxiliary tasks include POS tagging (Celano 2020; Arakelyan et al. 2018; Kondratyuk et al. 2018), morphological tagging (Arakelyan et al. 2018; Malaviya et al. 2019; Yildiz and Tantuğ 2019; Kondratyuk 2019; Zalmout and Habash 2019), and language modeling (Manjavacas et al. 2019). With a more linguistically-driven approach, Zalmout and Habash (2020) used subword modeling as an auxiliary task, and expanded the character-level representations to include the surrounding words of the focus word.

Table 13 shows a summary of all the introduced lemmatizers. Here we mark out the methods that utilize affix, syntactic, morphological, contextual, and character information as features, either handcrafted or automatically learned. Specifically, syntactic features refer to POS tags or coarse-grained POS categories. Contextual features are word *n*-grams or global information. Additionally, we indicate the mechanisms used by the lemmatizers to find the target lemma, namely, classification rule, edit script i.e. SES and edit tree, string similarity matching such as Levenshtein distance and vector shift, and character-level (or occasionally word-level) lemma generation. Among all, character-level generation garners the most interest due to the advancement of generative Seq2Seq models. Lastly, we list the lemmatization algorithms that adopt joint learning with other tasks, e.g., POS tagging, morphological tagging, and language modeling. The commonly used corpa are listed in Table 12. A comparison of performance can be found in Table 14.

With the help of neural networks and joint learning, lemmatization has achieved remarkable accuracy, especially for less inflection-rich languages such as English. The next step could be exploring the application of more recent pre-train language models such as GPT (Radford et al. 2018), or further improving multitask frameworks for joint learning. On the other hand, lemmatization for resource-scarce languages is still lacking. Although neural network approaches are proven to effective in ancient languages compared to a rule-based lemmatizer, for instance in Ancient Irish (Dereza 2018), there is still a lot of room for improvement. More research on unsupervised learning, semi-supervised learning, or transfer learning is called for resource-scarce language lemmatization.

Table 13 The comparison between different lemmatization methods

Method		Affix	Syntactic	Morph	Context	Char	CR	ES	SS	LG	Joint
Trans	Mladenic (2002)	✓			✓		✓				
	Plisson et al. (2004)	✓					✓				
	Juršič et al. (2007)	✓					✓				
	Erjavec and Džeroski (2004)	✓	✓				✓				
	Kanis and Müller (2005)	✓					✓				
	Jongejan and Dalianis (2009)	✓					✓				
	Daelemans et al. (2009)								✓		
	Gesmundo and Samardzic (2012)	✓	✓				✓				
Statistical	Chrupała (2006)				✓	✓		✓			
	Chrupala (2010)			✓		✓		✓			
	Müller et al. (2015)		✓	✓				✓			
	Lyras et al. (2007)	✓							✓		
	Dreyer et al. (2008)					✓			✓		✓
	Toutanova and Cherry (2009)	✓	✓			✓		✓			
	Nicolai and Kondrak (2016)	✓			✓	✓			✓		
	Barteld et al. (2016)		✓	✓		✓			✓		
	Gallay and Šimko (2016)	✓							✓		
	Rosa and Žabokrtský (2019)								✓		
	Akhmetov et al. (2020)					✓				✓	

Table 13 (continued)

Method		Affix	Syntactic	Morph	Context	Char	CR	ES	SS	LG	Joint
Neural	Kestemont et al. (2017)				✓	✓				✓	
	Chakrabarty et al. (2017)				✓	✓			✓		
	Bergmanis and Goldwater (2018)				✓	✓				✓	
	Bergmanis and Goldwater (2019)			✓	✓	✓				✓	✓
	Celano (2020)		✓							✓	✓
	Arakelyan et al. (2018)		✓	✓		✓				✓	✓
	Pütz et al. (2018)		✓	✓	✓					✓	
	Kondratyuk et al. (2018)		✓		✓	✓				✓	✓
	Malaviya et al. (2019)			✓	✓	✓				✓	✓
	Yildiz and Tantuğ (2019)			✓	✓	✓				✓	✓
	Manjavacas et al. (2019)				✓	✓				✓	✓
	Kondratyuk (2019)			✓	✓	✓			✓		✓
	Zalmout and Habash (2019)			✓		✓				✓	✓
	Chakrabarty et al. (2019)				✓	✓				✓	
	Schmitt and Constant (2019)		✓		✓	✓				✓	
	Zalmout and Habash (2020)				✓	✓				✓	✓
	Milintsevich and Sirts (2021)		✓		✓	✓				✓	

Trans is short for transformation. Statistical is short for statistical transduction, whereas Neural is short for neural transduction. Morph stands for morphological. Char stands for character. CR is classification rule. ES is edit script. SS is string similarity. LG is lemma generation. Joint is short for joint learning

Table 14 The performance of the introduced lemmatization methods

Method		Experiment 1				Experiment 2			
		L	D	R (%)	M	L	D	R	M
Transformation	Mladenic (2002)	Slovene	MULText-EAST	74.2	Acc.				
	Plisson et al. (2004)	Slovene	MULText-EAST	77.0	Acc.				
	Juršič et al. (2007)	Slovene	MULText-EAST	94.38	Acc.	English	Multext	94.14	Acc.
	Erjavec and Džeroski (2004)	Slovene	MULText-EAST	92.0	Acc.				
	Kanis and Müller (2005)	Czech	PDT	95.89	F1				
	Jongejan and Dalianis (2009)	Icelandic	IFD	71.3	Acc.	English	CELEX	89.0	Acc.
	Daelemans et al. (2009)	Afrikaans	D-Lem	91.0	Acc.				
	Gesmundo and Samardzic (2012)	English	*1984*	99.6	Acc.				
Statistical	Chrupała (2006)	Spanish	Cast3LB	92.48	F1	Catalan	Cast3LB	94.64	F1
	Chrupala (2010)	Spanish	CESS-ECE	98.52	Acc.	Romanian	MULText-EAST	97.78	Acc.
	Müller et al. (2015)	English	Pen Treebank	98.84	Acc.	Spanish	CoNLL-2009	98.46	Acc.
	Lyras et al. (2007)	Modern Greek	L-Lem	95.03	Acc.	English	L-Lem	96.46	Acc.
	Dreyer et al. (2008)	German	CELEX	94.0	Acc.				
	Toutanova and Cherry (2009)	English	CELEX	99.0	F1	Slovene	MULText-EAST	91.2	F1
	Nicolai and Kondrak (2016)	English	CoNLL-2009	93.3	Acc.	German	CELEX	90.0	Acc.
	Barteld et al. (2016)	German	GML	97.76	Acc.				
	Gallay and Šimko (2016)	MSA	PATB	95.4	Acc.	EGY	ARZ	83.3	Acc.
	Rosa and Žabokrtskỳ (2019)	English	UD v2.3[a]	97.78	Acc.				
	Akhmetov et al. (2020)	25 languages	Lem-list	84.05	avg Acc.				

Table 14 (continued)

Method		Experiment 1				Experiment 2			
		L	D	R (%)	M	L	D	R	M
Neural	Kestemont et al. (2017)	Middle Dutch	RELIG	90.97	Acc.	Middle Dutch	CG-LIT	91.67	Acc.
	Chakrabarty et al. (2017)	Hindi	WSD	94.90	Acc.	Spanish	CoNLL-2009	98.11	Acc.
	Bergmanis and Goldwater (2018)	20 languages	UD v2.0[b]	95.0	avg Acc.				
	Celano (2020)	Latin	LT4HALA	94.6	Acc.				
	Arakelyan et al. (2018)	English	CoNLL-2018	95.77	Acc.	French	CoNLL-2018	86.28	Acc.
	Pütz et al. (2018)	German	TüBa-D/Z	97.02	Acc.	German	NoSta-D	83.96	Acc.
	Kondratyuk et al. (2018)	Czech	PDT	98.37	Acc.	English	UD-EWT	97.53	Acc.
	Malaviya et al. (2019)	20 languages	UD v2.0	95.42	avg Acc.				
	Yildiz and Tantuğ (2019)	97 languages	UniMorph	97.85	avg Acc.				
	Manjavacas et al. (2019)	20 languages	UD v2.2[c]	96.28	avg Acc.				
	Kondratyuk (2019)	100 languages	SIGMORPHON	95.00	avg Acc.				
	Zalmout and Habash (2019)	MSA	PATB	97.6	Acc.	EGY	ARZ	88.5	Acc.
	Chakrabarty et al. (2019)	French	UD	95.47	Acc.	Hindi	UD	97.07	Acc.
	Schmitt and Constant (2019)	French	FTB	95.6	Acc.	Polish	SEJFEK	88.9	Acc.
	Zalmout and Habash (2020)	MSA	PATB	95.4	Acc.	EGY	ARZ	83.3	Acc.
	Milintsevich and Sirts (2021)	23 languages	UD v2.5[d]	97.09	avg Acc.				

L denotes language. D denotes dataset. R denotes result. M denotes measure. Statistical is short for statistical transduction. Neural is short for neural Transduction. MSA is Modern Standard Arabic. EGY is Egyptian Arabic

[a]http://hdl.handle.net/11234/1-2895

[b]http://hdl.handle.net/11234/1-1983

[c]http://hdl.handle.net/11234/1-2837

[d] http://hdl.handle.net/11234/1-3105

7 Conclusion

We believe that syntactic processing is a significant step in developing neurosymbolic AI and natural language understanding. As introduced in previous sections, there have been some attempts to use simple syntactic processing techniques to improve more complex NLP tasks. For instance, microtext normalization can highly improve sentiment analysis (Satapathy et al. 2017, 2019b). SBD can be incorporated as a sub-system in information retrieval (Krallinger et al. 2017). In sentiment analysis, it can help identify negation scope to improve the performance (Councill et al. 2010). POS tagging is used as part of the feature extraction process for cyber-bullying detection (Nandhini and Sheeba 2015) and query-based information retrieval (Mahmood et al. 2017). Asghar et al. (2014) suggested that POS tagging is also an important feature extraction step in sentiment analysis. Mubarok et al. (2017); Mao and Li (2021) applied it to aspect-based sentiment analysis and indeed obtained better results. Feng et al. (2019) proposed NMT decoding models that jointly predict target words and corresponding POS tag sequence, acquiring significant improvement. POS tagging is also used to provide syntactic structure in dialogue system (Meena et al. 2014; Bang et al. 2015). As for text chunking, Gupta et al. (2016) used both POS tagging and chunking as subtasks to tackle text summarization. Song et al. (2005) utilized chunking technique to improve biomedical information extraction. Syed et al. (2014) integrated text chunking with sentiment analysis in Urdu. Lemmatization has long been used for information retrieval (Halácsy and Trón 2006; Kanis and Skorkovská 2010; Balakrishnan and Lloyd-Yemoh 2014) and other tasks, e.g., text categorization (Camastra and Razi 2020), sentiment analysis (Mhatre et al. 2017), and Tweet stance classification (Priyanshu et al. 2020).

In this paper, we provide an extensive review on the above mentioned low-level syntactic processing techniques, summarizing different approaches, technical trends and challenges. We hope this survey can encourage more scholars to participate in the research of these fundamental techniques. We also hope this survey can inspire diverse neuro-symbolic AI systems to integrate these syntactic processing techniques in high-level NLP tasks in the future.

Acknowledgements This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project #A18A2b0046).

References

Abney S, Schapire RE, Singer Y (1999) Boosting applied to tagging and pp attachment. In: 1999 joint SIGDAT conference on empirical methods in natural language processing and very large corpora

Agarwal N, Ford KH, Shneider M (2005) Sentence boundary detection using a maxEnt classifier. In: Proceedings of MISC, pp 1–6

Aha DW, Kibler D, Albert MK (1991) Instance-based learning algorithms. Mach Learn 6(1):37–66

Akbik A, Blythe D, Vollgraf R (2018) Contextual string embeddings for sequence labeling. In: Proceedings of the 27th international conference on computational linguistics, pp 1638–1649

Akhmetov I, Pak A, Ualiyeva I, Gelbukh A (2020) Highly language-independent word lemmatization using a machine-learning classifier. Computación y Sistemas 24(3):1353

Altinok D (2018) An ontology-based dialogue management system for banking and finance dialogue systems. arXiv preprint arXiv:1804.04838

Altun Y, Tsochantaridis I, Hofmann T (2003) Hidden markov support vector machines. In: Proceedings of the 20th international conference on machine learning (ICML-03), pp 3–10

Alva P, Hegde V (2016) Hidden Markov model for POS tagging in word sense disambiguation. In: 2016 international conference on computation system and information technology for sustainable solutions (CSITSS). IEEE, pp 279–284

Ando RK, Zhang T (2005) A framework for learning predictive structures from multiple tasks and unlabeled data. J Mach Learn Res 6(Nov):1817–1853

Arakelyan G, Hambardzumyan K, Khachatrian H (2018) Towards jointud: part-of-speech tagging and lemmatization using recurrent neural networks. arXiv preprint arXiv:1809.03211

Asghar MZ, Khan A, Ahmad S, Kundi FM (2014) A review of feature extraction in sentiment analysis. J Basic Appl Sci Res 4(3):181–186

Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473

Balakrishnan V, Lloyd-Yemoh E (2014) Stemming and lemmatization: a comparison of retrieval performances. Lecture notes on software engineering, vol 2

Baldwin T, de Marneffe MC, Han B, Kim YB, Ritter A, Xu W (2015). Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In: Proceedings of the workshop on noisy user-generated text, Beijing, China. Association for Computational Linguistics, pp 126–135

Bang J, Noh H, Kim Y, Lee GG (2015) Example-based chat-oriented dialogue system with personalized long-term memory. In: 2015 international conference on big data and smart computing (BIGCOMP). IEEE, pp 238–243

Barteld F, Schröder I, Zinsmeister H (2016) Dealing with word-internal modification and spelling variation in data-driven lemmatization. In: Proceedings of the 10th SIGHUM workshop on language technology for cultural heritage, social sciences, and humanities, pp 52–62

Bartlett S, Kondrak G, Cherry C (2008) Automatic syllabification with structured SVMs for letter-to-phoneme conversion. In: Proceedings of ACL-08: HLT, pp 568–576

Beider A (2008) Beider-morse phonetic matching: an alternative to soundex with fewer false hits. Avotaynu 24(2):12

Belinkov Y, Màrquez L, Sajjad H, Durrani N, Dalvi F, Glass J (2018) Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. arXiv preprint arXiv:1801.07772

Bergmanis T, Goldwater S (2018) Context sensitive neural lemmatization with Lematus. In: Proceedings of the 2018 conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol 1 (Long Papers), pp 1391–1400

Bergmanis T, Goldwater S (2019) Training data augmentation for context-sensitive neural lemmatization using inflection tables and raw text. arXiv preprint arXiv:1904.01464

Biemann C (2006) Unsupervised part-of-speech tagging employing efficient graph clustering. In: Proceedings of the COLING/ACL 2006 student research workshop, pp 7–12

Böhmová A, Hajic J, Hajicová E, Hladká B, Abeillé A (2003) The prague dependency treebank: three-level annotation scenario. In: Treebanks: building and using parsed corpora, vol 20, pp 103–127

Bojanowski P, Grave E, Joulin A, Mikolov T (2016), 07. Enriching word vectors with subword information. Trans Assoc Comput Linguist. https://doi.org/10.1162/tacl_a_00051

Bontcheva K, Derczynski L, Funk A, Greenwood MA, Maynard D, Aswani N (2013) Twitie: An open-source information extraction pipeline for microblog text. Proceedings of the international conference recent advances in natural language processing RANLP 2013:83–90

Boudin F, Huet S, Torres-Moreno JM (2011) A graph-based approach to cross-language multi-document summarization. Polibits 43:113–118

Bouma G (2003) Finite state methods for hyphenation. Nat Lang Eng 9:5–20. https://doi.org/10.1017/S1351324903003073

Brants T (2000) TnT-a statistical part-of-speech tagger. arXiv preprint cs/0003055

Brill E (1995) Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. Comput Linguist 21(4):543–565

Brill E, Wu J (1998) Classifier combination for improved lexical disambiguation. In: 36th annual meeting of the Association for Computational Linguistics and 17th international conference on computational linguistics, vol 1, pp 191–195

Brody S, Diakopoulos N (2011) Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! using word lengthening to detect sentiment in microblogs. In: Proceedings of the 2011 conference on empirical methods in natural language processing, pp 562–570

Camastra F, Razi G (2020) Italian text categorization with lemmatization and support vector machines. In: Neural approaches to dynamics of signal exchanges. Springer, pp 47–54

Cambria E, Poria S, Gelbukh A, Thelwall M (2017) Sentiment analysis is a big suitcase. IEEE Intell Syst 32(6):74–80. https://doi.org/10.1109/MIS.2017.4531228

Cambria E, Liu Q, Decherchi S, Xing F, Kwok K (2022) Senticnet 7: a commonsense-based neurosymbolic AI framework for explainable sentiment analysis. Proceedings of LREC 2022:3829–3839

Cappé O, Godsill SJ, Moulines E (2007) An overview of existing methods and recent advances in sequential monte carlo. Proc IEEE 95(5):899–924

Cebron N, Berthold MR (2009) Active learning for object classification: from exploration to exploitation. Data Min Knowl Disc 18(2):283–299

Celano GG (2020) A gradient boosting-Seq2Seq system for Latin POS tagging and lemmatization. In: Proceedings of LT4HALA 2020-1st workshop on language technologies for historical and ancient languages, pp 119–123

Chakrabarty A, Pandit OA, Garain U (2017) Context sensitive lemmatization using two successive bidirectional gated recurrent networks. In: Proceedings of the 55th annual meeting of the Association for Computational Linguistics (vol 1: Long Papers), pp 1481–1491

Chakrabarty A, Chaturvedi A, Garain U (2019) CNN-based context sensitive lemmatization. In: Proceedings of the ACM India joint international conference on data science and management of data, CoDS-COMAD '19, New York, NY, USA. Association for Computing Machinery, pp 334–337

Chen T, Kan MY (2012) Creating a live, public short message service corpus: the nus sms corpus. Lang Resour Eval. https://doi.org/10.1007/s10579-012-9197-9

Chen D, Manning CD (2014) A fast and accurate dependency parser using neural networks. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 740–750

Chiu JP, Nichols E (2016) Named entity recognition with bidirectional LSTM-CNNs. Trans Assoc Comput Linguist 4:357–370

Choudhury M, Saraf R, Jain V, Mukherjee A, Sarkar S, Basu A (2007) Investigation and modeling of the structure of texting language. IJDAR 10(3):157–174. https://doi.org/10.1007/s10032-007-0054-0

Cho K, Van Merriënboer B, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: encoder-decoder approaches. arXiv preprint arXiv:1409.1259

Chrupała G (2006) Simple data-driven context-sensitive lemmatization. del Leng, Natural, Proces, p 37

Chrupala G (2010) Morfette: a tool for supervised learning of morphology

Chrupała G (2014) Normalizing tweets with edit scripts and recurrent neural embeddings. In: Proceedings of the 52nd annual meeting of the Association for Computational Linguistics (vol 2: Short Papers), pp 680–686

Chrupała G, Dinu G, Genabith J (2008), 01. Learning morphology with Morfette. In: Chrupała G, Dinu G, van Genabith J (2008) Learning morphology with Morfette. In: LREC 2008 - sixth international conference on language resources and evaluation, 28–30 May 2008, Marrakech, Morocco

Church KW (1988) A stochastic parts program and noun phrase parser for unrestricted text. Second conference on applied natural language processing. Austin, Texas, USA. Association for Computational Linguistics, pp 136–143

Civit M, Martí MA (2004) Building cast3lb: a Spanish treebank. Res Lang Comput 2(4):549–574

Clark A (2003) Combining distributional and morphological information for part of speech induction. In: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics—vol 1, EACL '03, USA. Association for Computational Linguistics, pp 59–66

Clark S, Curran JR, Osborne M (2003) Bootstrapping POS-taggers using unlabelled data. Proceedings of the seventh conference on natural language learning at HLT-NAACL 2003:49–55

Cohn DA, Ghahramani Z, Jordan MI (1996) Active learning with statistical models. J Artif Intell Res 4:129–145

Collins M (2002) Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In: Proceedings of the 2002 conference on empirical methods in natural language processing (EMNLP 2002). Association for Computational Linguistics, pp 1–8

Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12:2493–2537

Compton P, Edwards G, Kang B, Lazarus L, Malor R, Preston P, Srinivasan A (1992) Ripple down rules: turning knowledge acquisition into knowledge maintenance. Artif Intell Med 4(6):463–475

Councill I, McDonald R, Velikovich L (2010) What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In: Proceedings of the workshop on negation and speculation in natural language processing, pp 51–59

Curran JR, Clark S (2003) Investigating GIS and smoothing for maximum entropy taggers. In: 10th conference of the European chapter of the Association for Computational Linguistics

Daelemans W, Van den Bosch A, Weijters T (1997) Igtree: using trees for compression and classification in lazy learning algorithms. In: Lazy learning. Springer, pp 407–423

Daelemans W, Buchholz S, Veenstra J (1999a) Memory-based shallow parsing. arXiv preprint cs/9906005

Daelemans W, Zavrel J, Berck P, Gillis S (1999b) MBT: A memory-based part of speech tagger-generator. In: Fourth workshop on very large Corpora: 1996; Copenhagen, Denmark

Daelemans W, Zavrel J, van der Sloot K, van den Bosch A (2003) Timbl: Tilburg memory based learner, version 5.0, reference guide. Research Group Technical Report Series 3

Daelemans W, Zavrel J, Van Der Sloot K, Van den Bosch A (2004) Timbl: Tilburg memory-based learner. Tilburg University

Daelemans W, Groenewald HJ, van Huyssteen GB (2009) Prototype-based active learning for lemmatization. In: Proceedings of the international conference RANLP-2009, pp 65–70

Dai HJ, Lai PT, Chang YC, Tsai RTH (2015) Enhancing of chemical compound and drug name recognition using representative tag scheme and fine-grained tokenization. J Cheminform 7(1):1–10

Darwish K, Mubarak H, Abdelali A, Eldesouki M, Samih Y, Alharbi R, Attia M, Magdy W, Kallmeyer L (2018) Multi-dialect Arabic POS tagging: a CRF approach. In: Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)

Dauphin YN, Fan A, Auli M, Grangier D (2017) Language modeling with gated convolutional networks. In: International conference on machine learning. PMLR, pp 933–941

Dereza O (2018) Lemmatization for ancient languages: rules or neural networks? In: Ustalov D, Filchenkov A, Pivovarova L, Žižka J (eds) Artificial intelligence and natural language. Springer, Cham, pp 35–47

Desai N, Narvekar M (2015) Normalization of noisy text data. Procedia Comput Sci 45:127–132. https://doi.org/10.1016/j.procs.2015.03.104

Devlin J, Chang M, Lee K, Toutanova K (2018) BERT: pre-training of deep bidirectional transformers for language understanding. CoRR abs/1810.04805. arXiv:1810.04805

Dietterich TG, Bakiri G (1994) Solving multiclass learning problems via error-correcting output codes. J Artif Intell Res 2:263–286

Dipper S, Lüdeling A, Reznicek M (2013) Nosta-d: a corpus of german non-standard varieties. Non-standard data sources in corpus-based research 5:69–76

Dos Santos C, Zadrozny B (2014a) Learning character-level representations for part-of-speech tagging. In: International conference on machine learning. PMLR, pp 1818–1826

Dos Santos C, Zadrozny B (2014b) Learning character-level representations for part-of-speech tagging. In: International conference on machine learning. PMLR, pp 1818–1826

Dozat T, Manning CD (2016) Deep biaffine attention for neural dependency parsing. arXiv preprint arXiv:1611.01734

Dreyer M, Smith J, Eisner J (2008) Latent-variable modeling of string transductions with finite-state methods. In: Proceedings of the 2008 conference on empirical methods in natural language processing, pp 1080–1089

Dunlop M, Crossan A (2000) 08) Predictive text entry methods for mobile phones. Pers Technol. https://doi.org/10.1007/BF01324120

Dyer C, Ballesteros M, Ling W, Matthews A, Smith NA (2015) Transition-based dependency parsing with stack long short-term memory. arXiv preprint arXiv:1505.08075

Elman JL (1991) Distributed representations, simple recurrent networks, and grammatical structure. Mach Learn 7(2):195–225

Erjavec T (1998) The multext-east Slovene lexicon. In: Proceedings of the 7th electrotechnical conference ERK, vol B, pp 189–192

Erjavec T, Džeroski S, (2004) Machine learning of morphosyntactic structure: lemmatizing unknown slovene words. Appl Artif Intell 18(1):17–41

Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008) Liblinear: a library for large linear classification. J Mach Learn Res 9:1871–1874

Fatima M, Mueller MC (2019) HITS-SBD at the FinSBD task: machine learning vs. rule-based sentence boundary detection. In: Proceedings of the First workshop on financial technology and natural language processing, pp 115–121

Feng X, Feng Z, Zhao W, Zou N, Qin B, Liu T (2019) Improved neural machine translation with pos-tagging through joint decoding. In: International conference on artificial intelligence for communications and networks. Springer, pp 159–166

Forsyth EN (2007) Improving automated lexical and discourse analysis of online chat dialog. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA

Fossati D, Di Eugenio B (2008) I saw tree trees in the park: How to correct real-word spelling mistakes. In: LREC, pp 2008

Francis WN, Kucera H (1979) Brown corpus manual: manual of information to accompany a standard corpus of present-day edited american english for use with digital computers. Brown University, Providence, RI, USA

Freitag D, McCallum A (2000) Information extraction with hmm structures learned by stochastic optimization. AAAI/IAAI 2000:584–589

Gallay, L, Šimko M (2016) Utilizing vector models for automatic text lemmatization. In: International conference on current trends in theory and practice of informatics. Springer, pp 532–543

Ge M, Mao R, Cambria E (2022) Explainable metaphor identification inspired by conceptual metaphor theory. In: Proceedings of the 36th AAAI conference on artificial intelligence, pp 10681–10689

Gesmundo A, Samardzic T (2012) Lemmatisation as a tagging task. In: Proceedings of the 50th annual meeting of the Association for Computational Linguistics (vol 2: Short Papers), pp 368–372

Gillick D (2009) Sentence boundary detection and the problem with the US. In: Proceedings of human language technologies: the 2009 annual conference of the North American chapter of the Association for Computational Linguistics, companion volume: Short Papers, pp 241–244

Giménez J, Marquez L (2004a) Fast and accurate part-of-speech tagging: the SVM approach revisited. Rec Adv Nat Lang Process III:153–162

Giménez J, Màrquez L (2004b) SVMTool: A general POS tagger generator based on support vector machines. In: Proceedings of the fourth international conference on language resources and evaluation (LREC'04), Lisbon, Portugal. European Language Resources Association (ELRA)

Gimpel K, Schneider N, O'Connor B, Das D, Mills D, Eisenstein J, Heilman M, Yogatama D, Flanigan J, Smith NA (2010) Part-of-speech tagging for twitter: annotation, features, and experiments. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science

Gotoh Y, Renals S (2000) Sentence boundary detection in broadcast speech transcripts. In: ASR2000-automatic speech recognition: challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)

Graves A, Mohamed Ar, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing. IEEE, pp 6645–6649

Grefenstette G, Tapanainen P (1994) What is a word, what is a sentence?: Problems of tokenisation. Report, Grenoble Laboratory

Griffis D, Shivade C, Fosler-Lussier E (2016) Lai AM (2016) A quantitative and qualitative evaluation of sentence boundary detection for the clinical domain. AMIA Jt Summits on Transl Sci Proc 27570656:88–97

Grove AJ, Roth D (2001) Linear concepts and hidden variables. Mach Learn 42(1–2):123–141

Gui T, Zhang Q, Huang H, Peng M, Huang X (2017), September. Part-of-speech tagging for twitter with adversarial neural networks. In: Proceedings of the 2017 conference on empirical methods in natural language processing, Copenhagen, Denmark. Association for Computational Linguistics, pp 2411–2420

Gupta H, Kottwani A, Gogia S, Chaudhari S (2016) Text analysis and information retrieval of text data. In: 2016 international conference on wireless communications, signal processing and networking (WiSPNET). IEEE, pp 788–792

Hajič J, Ciaramita M, Johansson R, Kawahara D, Martí MA, Màrquez L, Meyers A, Nivre J, Padó S, Štěpánek J, Straňák P, Surdeanu M, Xue N, Zhang Y (2009) The CoNLL-2009 shared task: syntactic and semantic dependencies in multiple languages. In: Proceedings of the thirteenth conference on computational natural language learning: shared task, CoNLL '09, USA. Association for Computational Linguistics, pp 1–18

Halácsy P, Trón V (2006) Benefits of deep NLP-based lemmatization for information retrieval. Citeseer, In CLEF (Working Notes)

Han B, Baldwin T (2011a) Lexical normalisation of short text messages: Makn sens a# twitter. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies, pp 368–378

Han B, Baldwin T (2011b) Lexical normalisation of short text messages: Makn sens a #twitter. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies, Portland, Oregon, USA. Association for Computational Linguistics, pp 368–378

Hart PE, Nilsson NJ, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern 4(2):100–107

Helgadóttir S (2012) Icelandic frequency dictionary 2012.11-training/testing sets

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780

How Y, yen Kan M (2005) Optimizing predictive text entry for short message service on mobile phones. In: In human computer interfaces international (HCII 05). 2005: Las Vegas

Huang Z, Xu W, Yu K (2015) Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991

Ide N, Véronis J (1994) Multext: multilingual text tools and corpora. In: COLING 1994 vol 1: the 15th international conference on computational linguistics

Jahjah V, Khoury R, Lamontagne L (2016) Word normalization using phonetic signatures. In: Canadian conference on artificial intelligence. Springer, pp 180–185

Jiampojamarn S, Kondrak G, Sherif T (2007) Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In: Human language technologies 2007: the conference of the North American chapter of the Association for Computational Linguistics; proceedings of the main conference, pp 372–379

Jiampojamarn S, Cherry C, Kondrak G (2010) Integrating joint n-gram features into a discriminative training framework. Human language technologies: the 2010 annual conference of the North American chapter of the Association for Computational Linguistics. Los Angeles, California. Association for Computational Linguistics, pp 697–700

Jing H, Lopresti D, Shih C (2003) Summarization of noisy documents: a pilot study. In: Proceedings of the HLT-NAACL 03 text summarization workshop, pp 25–32

Joachims T (2002) Optimizing search engines using clickthrough data. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, pp 133–142

Johnson R, Zhang T (2005) A high-performance semi-supervised learning method for text chunking. In: Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL'05), pp 1–9

Johnson AE, Pollard TJ, Shen L, Li-Wei HL, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, Mark RG (2016) Mimic-iii, a freely accessible critical care database. Sci Data 3(1):1–9

Jones KS (1972) A statistical interpretation of term specificity and its application in retrieval. J Doc 28:11–21

Jongejan B, Dalianis H (2009) Automatic training of lemmatization rules that handle morphological changes in pre-, in-and suffixes alike. In: Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP, pp 145–153

Jose G, Raj NS (2014) Lexico-syntactic normalization model for noisy SMS text. In: 2014 international conference on electronics, communication and computational engineering (ICECCE). IEEE, pp 163–168

Judge J, Cahill A, van Genabith J (2006) QuestionBank: creating a corpus of parse-annotated questions. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the Association for Computational Linguistics, Sydney, Australia. Association for Computational Linguistics, pp 497–504

Juršič M, Mozetič I, Lavrač N (2007) Learning ripple down rules for efficient lemmatization. In: Proceedings of the 10th international multiconference information society, IS, pp 206–209

Kanakaraddi SG, Nandyal SS (2018) Survey on parts of speech tagger techniques. In: 2018 international conference on current trends towards converging technologies (ICCTCT). IEEE, pp 1–6

Kanis J, Müller L (2005) Automatic lemmatizer construction with focus on OOV words lemmatization. In: International Conference on Text, Speech and Dialogue, pp 132–139. Springer

Kanis J, Skorkovská L (2010) Comparison of different lemmatization approaches through the means of information retrieval performance. In: Sojka P, Horák A, Kopeček I, Pala K (eds) Text, Speech and Dialogue, Berlin, Heidelberg. Springer, Berlin Heidelberg, pp 93–100

Kaufmann M, Kalita J (2010) Syntactic normalization of twitter messages. In: International conference on natural language processing, Kharagpur, India, Volume, p 16

Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY (2017) Lightgbm: A highly efficient gradient boosting decision tree. Adv Neural Inf Process Syst 30:3146–3154

Kestemont M, De Pauw G, van Nie R, Daelemans W (2017) Lemmatization for variation-rich languages using deep learning. Digital Scholarship in the Humanities 32(4):797–815

Khapra M, Kulkarni A, Sohoney S, Bhattacharyya P (2010) All words domain adapted WSD: Finding a middle ground between supervision and unsupervision. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, pp 1532–1541. Association for Computational Linguistics

Khoury R (2015) Phonetic normalization of microtext. In: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015, pp 1600–1601

Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp 1746–1751. Association for Computational Linguistics

Kim JD, Ohta T, Tateisi Y, Tsujii J (2003) Genia corpus—a semantically annotated corpus for bio-textmining. Bioinformatics 19(suppl-1):i180–i182

Kim Y, Jernite Y, Sontag D, Rush AM (2016) Character-aware neural language models. In: Thirtieth AAAI conference on artificial intelligence

Kirov C, Cotterell R, Sylak-Glassman J, Walther G, Vylomova E, Xia P, Faruqui M, Mielke SJ, McCarthy AD, Kübler S, et al. (2018) Unimorph 2.0: universal morphology. arXiv preprint arXiv:1810.11101

Kiss T, Strunk J (2002) Viewing sentence boundary detection as collocation identification. In: Proceedings of KONVENS, vol 2002. Citeseer, pp 75–82

Kiss T, Strunk J (2006) 12. Unsupervised multilingual sentence boundary detection. Comput Linguist 32(4):485–525. https://doi.org/10.1162/coli.2006.32.4.485. https://direct.mit.edu/coli/article-pdf/32/4/485/1798345/coli.2006.32.4.485.pdf

Klambauer G, Unterthiner T, Mayr A, Hochreiter S (2017) Self-normalizing neural networks. In: Proceedings of the 31st international conference on neural information processing systems, pp 972–981

Knoll BC, Lindemann EA, Albert AL, Melton GB, Pakhomov SVS (2019) Recurrent deep network models for clinical nlp tasks: Use case with sentence boundary disambiguation. Studies in health technology and informatics 264(31437913):198–202. https://doi.org/10.3233/SHTI190211

Koehn P, Hoang H, Birch A, Callison-Burch C, Federico M, Bertoldi N, Cowan B, Shen W, Moran C, Zens R et al (2007) Moses: Open source toolkit for statistical machine translation. In: Proceedings of the 45th annual meeting of the Association for Computational Linguistics companion volume proceedings of the demo and poster sessions, pp 177–180

Koeling R (2000) Chunking with maximum entropy models. In: Fourth conference on computational natural language learning and the second learning language in logic workshop

Kondratyuk D (2019) Cross-lingual lemmatization and morphology tagging with two-stage multilingual BERT fine-tuning. In: Proceedings of the 16th workshop on computational research in phonetics, phonology, and morphology, pp 12–18

Kondratyuk D, Straka M (2019) 75 languages, 1 model: parsing universal dependencies universally. arXiv preprint arXiv:1904.02099

Kondratyuk D, Gavenčiak T, Straka M, Hajič J (2018) Lemmatag: jointly tagging and lemmatizing for morphologically-rich languages with BRNNs. arXiv preprint arXiv:1808.03703

Krallinger M, Rabal O, Lourenco A, Oyarzabal J, Valencia A (2017) Information retrieval and text mining technologies for chemistry. Chem Rev 117(12):7673–7761

Kudo T, Matsumoto Y (2000) Use of support vector learning for chunk identification. In: Fourth conference on computational natural language learning and the second learning language in logic workshop

Kudo T, Matsumoto Y (2001) Chunking with support vector machines. In: Second meeting of the North American chapter of the Association for Computational Linguistics

Kupiec J (1992) Robust part-of-speech tagging using a hidden markov model. Comput Speech Lang 6(3):225–242. https://doi.org/10.1016/0885-2308(92)90019-Z

Kuru O, Can OA, Yuret D (2016) CharNER: character-level named entity recognition. In: Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical Papers, Osaka, Japan. The COLING 2016 Organizing Committee, pp 911–921

Lafferty JD, McCallum A, Pereira FCN (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the eighteenth international conference on machine learning, ICML '01, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc, pp 282–289

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324

Lee YS, Wu YC (2007) A robust multilingual portable phrase chunking system. Expert Syst Appl 33(3):590–599

Leeman-Munk S, Lester J, Cox J (2015) Ncsu_sas_sam: Deep encoding and reconstruction for normalization of noisy text. In: Proceedings of the workshop on noisy user-generated text, pp 154–161

Li C, Liu Y (2012) Normalization of text messages using character-and phone-based machine translation approaches. In: Thirteenth annual conference of the international speech communication association

Lin JCW, Shao Y, Zhang J, Yun U (2020) Enhanced sequence labeling based on latent variable conditional random fields. Neurocomputing 403:431–440

Ling W, Dyer C, Black AW, Trancoso I (2015a) Two/too simple adaptations of word2vec for syntax problems. In: Proceedings of the 2015 conference of the North American chapter of the Association for Computational Linguistics: Human Language Technologies, pp 1299–1304

Ling W, Luís T, Marujo L, Astudillo RF, Amir S, Dyer C, Black AW, Trancoso I (2015b) Finding function in form: compositional character models for open vocabulary word representation. arXiv preprint arXiv:1508.02096

Littlestone N (1988) Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. Mach Learn 2(4):285–318

Liu Y, Stolcke A, Shriberg E, Harper M (2004) Comparing and combining generative and posterior probability models: some advances in sentence boundary detection in speech. In: Proceedings of the 2004 conference on empirical methods in natural language processing, pp 64–71

Liu Y, Stolcke A, Shriberg E, Harper M (2005) Using conditional random fields for sentence boundary detection in speech. In: Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL'05), pp 451–458

Liu Y, Chawla NV, Harper MP, Shriberg E, Stolcke A (2006) A study in machine learning from imbalanced data for sentence boundary detection in speech. Comput Speech Lang 20(4):468–494. https://doi.org/10.1016/j.csl.2005.06.002

Liu F, Weng F, Wang B, Liu Y (2011) Insertion, deletion, or substitution? normalizing text messages without pre-categorization nor supervision. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, Oregon, USA. Association for Computational Linguistics, pp 71–76

Liu F, Weng F, Jiang X (2012) A broad-coverage normalization system for social media language. In: Proceedings of the 50th annual meeting of the association for computational linguistics (vol 1: Long Papers), pp 1035–1044

Liu H, Dacon J, Fan W, Liu H, Liu Z, Tang J (2019) Does gender matter? towards fairness in dialogue systems. arXiv preprint arXiv:1910.10486

Liu Y, Li G, Zhang X (2020) Semi-Markov CRF model based on stacked neural Bi-LSTM for sequence labeling. In: 2020 IEEE 3rd international conference of safe production and informatization (IICSPI), pp 19–23

Lourentzou I, Manghnani K, Zhai C (2019) Adapting sequence to sequence models for text normalization in social media. Proceedings of the international AAAI conference on web and social media, vol 13, pp 335–345

Luhn HP (1957) A statistical approach to mechanized encoding and searching of literary information. IBM J Res Dev 1(4):309–317

Luong MT, Pham H, Manning CD (2015) Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025

Lusetti M, Ruzsics T, Göhring A, Samardzic T, Stark E (2018) Encoder-decoder methods for text normalization. In: VarDial@COLING 2018

Lyras DP, Sgarbas KN, Fakotakis ND (2007) Using the Levenshtein edit distance for automatic lemmatization: a case study for modern Greek and English. In: 19th IEEE international conference on tools with artificial intelligence (ICTAI 2007), vol 2. IEEE, pp 428–435

Ma X, Hovy E (2016) End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. arXiv preprint arXiv:1603.01354

Ma J, Zhu J, Xiao T, Yang N (2013) Easy-first POS tagging and dependency parsing with beam search. In: Proceedings of the 51st annual meeting of the Association for Computational Linguistics (vol 2: Short Papers), pp 110–114

Maamouri M, Bies A, Buckwalter T, Mekki W (2004) The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus. In: NEMLAR conference on Arabic language resources and tools, vol 27. Cairo, pp 466–467

Maamouri M, Krouna S, Tabessi D, Hamrouni N, Habash N (2012) Egyptian Arabic morphological annotation guidelines

Mahmood A, Khan HU, ur Rehman Z, Khan W (2017) Query based information retrieval and knowledge extraction using hadith datasets. In: 2017 13th international conference on emerging technologies (ICET), pp 1–6

Malaviya C, Wu S, Cotterell R (2019) A simple joint model for improved contextual neural lemmatization. arXiv preprint arXiv:1904.02306

Manandhar S, Džeroski S, Erjavec T (1998) Learning multilingual morphology with clog. In: International conference on inductive logic programming. Springer, pp 135–144

Manjavacas E, Kádár Á, Kestemont M (2019) Improving lemmatization of non-standard languages with joint learning. arXiv preprint arXiv:1903.06939

Manning CD (2011) Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In: Gelbukh AF (ed) Computational linguistics and intelligent text processing. Springer, Berlin, pp 171–189

Mao R, Li X (2021) Bridging towers of multi-task learning with a gating mechanism for aspect-based sentiment analysis and sequential metaphor identification. In: Proceedings of the AAAI conference on artificial intelligence, vol 35, pp 13534–13542

Mao R, Lin C, Guerin F (2018) Word embedding and WordNet based metaphor identification and interpretation. In: Proceedings of the 56th annual meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 1222–1231

Mao R, Lin C, Guerin F (2019) End-to-end sequential metaphor identification inspired by linguistic theories. In: Proceedings of the 57th annual meeting of the Association for Computational Linguistics (Long Papers), pp 3888–3898

Mao R, Lin C, Guerin F (2021) Combining pre-trained word embeddings and linguistic features for sequential metaphor identification. arXiv preprint arXiv:2104.03285

Mao R, Li X, Ge M, Cambria E (2022) Metapro: A computational metaphor processing model for text pre-processing. Inf Fus 86–87:30–43. https://doi.org/10.1016/j.inffus.2022.06.002

Marcus MP, Santorini B, Marcinkiewicz MA (1993) Building a large annotated corpus of English: The Penn Treebank. Comput Linguist 19(2):313–330

Martı MA, Taulé M, Márquez L, Bertran M (2007) CESS-ECE: a multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/mbertran/cess-ece

Matsoukas S, Bulyko I, Xiang B, Nguyen K, Schwartz R, Makhoul J (2007) Integrating speech recognition and machine translation. In: 2007 IEEE international conference on acoustics, speech and signal processing-ICASSP'07, vol 4. IEEE, pp IV–1281

McCallum A, Freitag D, Pereira FC (2000) Maximum entropy markov models for information extraction and segmentation. Icml 17:591–598

McCarthy AD, Vylomova E, Wu S, Malaviya C, Wolf-Sonkin L, Nicolai G, Kirov C, Silfverberg M, Mielke SJ, Heinz J, et al. (2019) The SIGMORPHON 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. arXiv preprint arXiv:1910.11493

McCord MC (1990) Slot grammar, natural language and logic. Springer, pp 118–145

McDonald R, Crammer K, Pereira F (2005) Flexible text segmentation with structured multilabel classification. In: Proceedings of human language technology conference and conference on empirical methods in natural language processing, pp 987–994

Meena R, Skantze G, Gustafson J (2014) Data-driven models for timing feedback responses in a map task dialogue system. Comput Speech Lang 28(4):903–922

Meftah S, Semmar N (2018) A neural network model for part-of-speech tagging of social media texts. In: Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018), Miyazaki, Japan. European Language Resources Association (ELRA)

Mhatre M, Phondekar D, Kadam P, Chawathe A, Ghag K (2017) Dimensionality reduction for sentiment analysis using pre-processing techniques. In: 2017 international conference on computing methodologies and communication (ICCMC). IEEE, pp 16–21

Mikheev A (2000) Tagging sentence boundaries. In: 1st meeting of the North American chapter of the Association for Computational Linguistics

Mikheev A (2002) Periods, capitalized words, etc. Comput Linguist 28(3):289–318

Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: Proceedings of workshop at ICLR 2013

Milintsevich K, Sirts K (2021) Enhancing sequence-to-sequence neural lemmatization with external resources. arXiv preprint arXiv:2101.12056

Mittal A, Bhatt P, Kumar P (2014) Phonetic matching and syntactic tree similarity based QA system for SMS queries. In: 2014 international conference on green computing communication and electrical engineering (ICGCCEE). IEEE, pp 1–6

Mladenic D (2002) Automatic word lemmatization. In: Proceedings of the 5th international multi-conference information society, IS-2002 B, pp 153–159

Molina A, Pla F (2002) Shallow parsing using specialized HMMs. J Mach Learn Res 2(Mar):595–613

Morency LP, Quattoni A, Darrell T (2007) Latent-dynamic discriminative models for continuous gesture recognition. In: 2007 IEEE conference on computer vision and pattern recognition, pp 1–8

Mubarok MS, Adiwijaya, Aldhi MD (2017) Aspect-based sentiment analysis to review products using Naïve Bayes. AIP Conf Proc 1867:020060

Muis AO, Lu W (2016) Weak semi-Markov CRFs for noun phrase chunking in informal text. In: Proceedings of the 2016 conference of the North American chapter of the Association for Computational Linguistics: Human Language Technologies, pp 714–719

Müller T, Schmid H, Schütze H (2013) Efficient higher-order crfs for morphological tagging. In: Proceedings of the 2013 conference on empirical methods in natural language processing, pp 322–332

Müller T, Cotterell R, Fraser A, Schütze H (2015) Joint lemmatization and morphological tagging with Lemming. In: Proceedings of the 2015 conference on empirical methods in natural language processing, pp 2268–2274

Nakagawa T, Kudo T, Matsumoto Y (2001) Unknown word guessing and part-of-speech tagging using support vector machines. In: NLPRS. Citeseer, pp 325–331

Nandhini BS, Sheeba J (2015) Cyberbullying detection and classification using information retrieval algorithm. In: Proceedings of the 2015 international conference on advanced research in computer science engineering & technology (ICARCSET 2015), pp 1–5

Ngai G, Florian R (2001) Transformation based learning in the fast lane. In: Second meeting of the North American chapter of the Association for Computational Linguistics

Nguyen DQ, Vu T, Nguyen DQ, Dras M, Johnson M (2017) From word segmentation to POS tagging for Vietnamese. arXiv preprint arXiv:1711.04951

Nicolai G, Kondrak G (2016) Leveraging inflection tables for stemming and lemmatization. In: Proceedings of the 54th annual meeting of the Association for Computational Linguistics (vol 1: Long Papers), pp 1138–1147

Niehues J, Cho E (2017) Exploiting linguistic resources for neural machine translation using multi-task learning. arXiv preprint arXiv:1708.00993

Nivre J, Hall J, Kübler S, McDonald R, Nilsson J, Riedel S, Yuret D (2007) The CoNLL 2007 shared task on dependency parsing. In: Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL), Prague, Czech Republic. Association for Computational Linguistics, pp 915–932

Nivre J, De Marneffe MC, Ginter F, Goldberg Y, Hajic J, Manning CD, McDonald R, Petrov S, Pyysalo S, Silveira N, et al. (2016) Universal dependencies v1: a multilingual treebank collection. In: Proceedings of the tenth international conference on language resources and evaluation (LREC'16), pp 1659–1666

Nocedal J, Wright S (2006) Numerical optimization. Springer, New York

O'Connor B, Krieger M, Ahn D (2010) Tweetmotif: exploratory search and topic summarization for twitter. In: Fourth international AAAI conference on weblogs and social media

Otter DW, Medina JR, Kalita JK (2020) A survey of the usages of deep learning for natural language processing. IEEE transactions on neural networks and learning systems 32(2):604–624

Owoputi O, O'Connor B, Dyer C, Gimpel K, Schneider N, Smith NA (2013) Improved part-of-speech tagging for online conversational text with word clusters. In: Proceedings of the 2013 conference of the North American chapter of the Association for Computational Linguistics: human language technologies, pp 380–390

Palmer DD, Hearst MA (1994) Adaptive sentence boundary disambiguation. arXiv preprint cmp-lg/9411022

Palmer DD, Hearst MA (1997) Adaptive multilingual sentence boundary disambiguation. Comput. Linguist. 23(2):241–267

Papineni K, Roukos S, Ward T, Zhu WJ (2002) Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pp 311–318

Pennell DL, Liu Y (2010) Normalization of text messages for text-to-speech. In: 2010 IEEE international conference on acoustics, speech and signal processing. IEEE, pp 4842–4845

Pennell D, Liu Y (2011) A character-level machine translation approach for normalization of SMS abbreviations. In: Proceedings of 5th international joint conference on natural language processing, pp 974–982

Pennell DL, Liu Y (2014) Normalization of informal text. Comput Speech Lang 28(1):256–277

Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543

Peters R, Nagel N (2014) Das digitale, referenzkorpus mittelniederdeutsch/niederrheinisch (ReN)'. Jahrbuch für Germanistische Sprachgeschichte 5(1):165–175

Petrov S, Das D, McDonald R (2011) A universal part-of-speech tagset. arXiv preprint arXiv:1104.2086

Petrović S, Osborne M, Lavrenko V (2010) The Edinburgh twitter corpus. In: Proceedings of the NAACL HLT 2010 workshop on computational linguistics in a world of social media, pp 25–26

Philips L (1990) Hanging on the metaphone. Comput Lang 7(12):39–43

Philips L (2000) The double metaphone search algorithm. C/C++ users J 18(6):38–43

Platt J (1999) Fast training of support vector machines using sequential minimal optimization. Adv Kernel Methods 185–208

Plisson J, Lavrac N, Mladenic D et al (2004) A rule based approach to word lemmatization. Proceedings of IS 3:83–86

Priyanshu A, Das VR, Rajiv Moghe S, Rathod H, Medicherla SS, Shail Chhabra M, Shastri S (2020) Stance classification with improved elementary classifiers using lemmatization (grand challenge). In: 2020 IEEE sixth international conference on multimedia big data (BigMM), pp 466–470

Punyakanok V, Roth D (2000) The use of classifiers in sequential inference. Adv Neural Inf Process Syst 13:995–1001

Pütz T, De Kok D, Pütz S, Hinrichs E (2018) Seq2seq or perceptrons for robust lemmatization. an empirical examination. In: Proceedings of the 17th international workshop on treebanks and linguistic theories (TLT 2018), pp 193–207

Qi P, Dozat T, Zhang Y, Manning CD (2018) Universal Dependency parsing from scratch. In: Proceedings of the CoNLL 2018 shared task: multilingual parsing from raw text to universal dependencies, Brussels, Belgium. Association for Computational Linguistics, pp 160–170

Qi P, Zhang Y, Zhang Y, Bolton J, Manning CD (2020) Stanza: a python natural language processing toolkit for many human languages. arXiv preprint arXiv:2003.07082

Rabiner LR (1989) A tutorial on hidden markov models and selected applications in speech recognition. Proc IEEE 77(2):257–286

Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding by generative pre-training. Report, OpenAI

Raffel C, Luong MT, Liu PJ, Weiss RJ, Eck D (2017) Online and linear-time attention by enforcing monotonic alignments. In: International conference on machine learning, pp 2837–2846. PMLR

Ramshaw LA, Marcus MP (1999) Text chunking using transformation-based learning, Natural language processing using very large corpora, 157–176. Springer

Rastogi P, Cotterell R, Eisner J (2016) Weighting finite-state transductions with neural context. In: Proceedings of the 2016 conference of the North American chapter of the Association for Computational Linguistics: human language technologies, pp 623–633

Ratinov L, Roth D (2009) Design challenges and misconceptions in named entity recognition. In: Proceedings of the thirteenth conference on computational natural language learning (CoNLL-2009), pp 147–155

Ratnaparkhi A (1996) A maximum entropy model for part-of-speech tagging. In: Conference on empirical methods in natural language processing

Read J, Dridan R, Oepen S, Solberg LJ (2012) Sentence boundary detection: a long solved problem? In Proceedings of COLING 2012: posters, pp 985–994

Rei M (2017) Semi-supervised multitask learning for sequence labeling. arXiv preprint arXiv:1704.07156

Reynar JC, Ratnaparkhi A (1997) A maximum entropy approach to identifying sentence boundaries. arXiv preprint cmp-lg/9704002

Riley M (1989) Some applications of tree-based modelling to speech and language. In: Speech and natural language: proceedings of a workshop held at Cape Cod, Massachusetts, October 15–18

Ritter A, Clark S, Etzioni O, et al. (2011) Named entity recognition in tweets: an experimental study. In: Proceedings of the 2011 conference on empirical methods in natural language processing, pp 1524–1534

Roberts L (2016) Syntactic processing. Cambridge handbooks in language and linguistics. Cambridge University Press, Cambridge, pp 227–247

Rosa KD, Ellen J (2009) Text classification methodologies applied to micro-text in military chat. In: 2009 international conference on machine learning and applications, pp 710–714

Rosa R, Žabokrtskỳ Z (2019) Unsupervised lemmatization as embeddings-based word clustering. arXiv preprint arXiv:1908.08528

Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev 65(6):386

Rudrapal D, Jamatia A, Chakma K, Das A, Gambäck B (2015) Sentence boundary detection for social media text. In: Proceedings of the 12th international conference on natural language processing, Trivandrum, India. NLP Association of India, pp 254–260

Rush AM, Reichart R, Collins M, Globerson A (2012) Improved parsing and POS tagging using inter-sentence consistency constraints. In: Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning, pp 1434–1444

Ruzsics T, Samardzic T (2017) Neural sequence-to-sequence learning of internal word structure. In: Proceedings of the 21st conference on computational natural language learning (CoNLL 2017), pp 184–194

Sadvilkar N, Neumann M (2020) PySBD: pragmatic sentence boundary disambiguation. arXiv preprint arXiv:2010.09657

Salzberg SL (1994) C4.5: Programs for machine learning by J. Ross Quinlan. Morgan Kaufmann Publishers, inc., 1993. Mach Learn 16(3): 235–240. https://doi.org/10.1007/BF00993309

Sanchez G (2019) Sentence boundary detection in legal text. Proceedings of the natural legal language processing workshop 2019:31–38

Sang ETK (2000) Text chunking by system combination. In: Fourth conference on computational natural language learning and the second learning language in logic workshop

Sang EF, Buchholz S (2000) Introduction to the CoNLL-2000 shared task: Chunking. arXiv preprint cs/0009008

Sang EF, Veenstra J (1999) Representing text chunks. arXiv preprint cs/9907006

Santos CNd, Guimaraes V (2015) Boosting named entity recognition with neural character embeddings. arXiv preprint arXiv:1505.05008

Sarawagi S, Cohen WW (2004) Semi-Markov conditional random fields for information extraction. Adv Neural Inf Process Syst 17:1185–1192

Satapathy R, Guerreiro C, Chaturvedi I, Cambria E (2017) Phonetic-based microtext normalization for twitter sentiment analysis. In: 2017 IEEE international conference on data mining workshops (ICDMW), pp 407–413

Satapathy R, Li Y, Cavallari S, Cambria E (2019a) Seq2seq deep learning models for microtext normalization. In: 2019 international joint conference on neural networks (IJCNN), pp 1–8. IEEE

Satapathy R, Singh A, Cambria E (2019b) Phonsenticnet: a cognitive approach to microtext normalization for concept-level sentiment analysis. In: International conference on computational data and social networks, pp 177–188. Springer

Satapathy R, Cambria E, Nanetti A, Hussain A (2020) A review of shorthand systems: from brachygraphy to microtext and beyond. Cogn Comput 12(4):778–792

Savary A, Zaborowski B, Krawczyk-Wieczorek A, Makowiecki F (2012) Sejfek - a lexicon and a shallow grammar of polish economic multi-word units. In: Proceedings of the 3rd workshop on cognitive aspects of the lexicon, pp 195–214

Schapire RE, Singer Y (1999) Improved boosting algorithms using confidence-rated predictions. Mach Learn 37(3):297–336

Schmid H (2000) Unsupervised learning of period disambiguation for tokenisation. Internal Report, IMS-CL

Schmitt M, Constant M (2019) Neural lemmatization of multiword expressions. In: Proceedings of the joint workshop on multiword expressions and wordnet (MWE-WN 2019), pp 142–148

Seddah D, Tsarfaty R, Kübler S, Candito M, Choi J, Farkas R, Foster J, Goenaga I, Gojenola K, Goldberg Y et al. (2013) Overview of the SPMRL 2013 shared task: cross-framework evaluation of parsing morphologically rich languages. In: Proceedings of the fourth workshop on statistical parsing of morphologically-rich languages. Association for Computational Linguistics

Sennrich R, Firat O, Cho K, Birch A, Haddow B, Hitschler J, Junczys-Dowmunt M, Läubli S, Barone AVM, Mokry J et al. (2017) Nematus: a toolkit for neural machine translation. arXiv preprint arXiv:1703.04357

Sha F, Pereira F (2003) Shallow parsing with conditional random fields. In: Proceedings of the 2003 human language technology conference of the North American Chapter of the Association for Computational Linguistics, pp 213–220

Shao Y, Hardmeier C, Tiedemann J, Nivre J (2017) Character-based joint segmentation and pos tagging for chinese using bidirectional rnn-crf. arXiv preprint arXiv:1704.01314

Shen L, Satta G, Joshi A (2007) Guided learning for bidirectional sequence classification. In: Proceedings of the 45th annual meeting of the Association of Computational Linguistics, Prague, Czech Republic. Association for Computational Linguistics, pp 760–767

Shewchuk JR et al. (1994) An introduction to the conjugate gradient method without the agonizing pain

Silveira N, Dozat T, De Marneffe MC, Bowman SR, Connor M, Bauer J, Manning CD (2014) A gold standard dependency corpus for English. In: LREC, pp 2897–2904. Citeseer

Skut W, Krenn B, Brants T, Uszkoreit H (2002) 05. An annotation scheme for free word order languages. Proceedings of the 5th conference on applied natural language processing. https://doi.org/10.3115/974557.974571

Søgaard A (2010) Simple semi-supervised training of part-of-speech taggers. In: Proceedings of the ACL 2010 conference short papers, Uppsala, Sweden. Association for Computational Linguistics, pp 205–208

Song M, Song IY, Hu X, Allen RB (2005) Integrating text chunking with mixture hidden Markov models for effective biomedical information extraction. In: International conference on computational science. Springer, pp 976–984

Spoustová Dj, Hajič J, Raab J, Spousta M (2009) Semi-supervised training for the averaged perceptron POS tagger. In: Proceedings of the 12th conference of the European chapter of the ACL (EACL 2009), Athens, Greece. Association for Computational Linguistics, pp 763–771

Stamatatos E, Fakotakis N, Kokkinakis G (1999) Automatic extraction of rules for sentence boundary disambiguation. In: Proceedings of the workshop on machine learning in human language technology. Citeseer, pp 88–92

Stevenson M, Gaizauskas R (2000) Experiments on sentence boundary detection. In: Sixth applied natural language processing conference, pp 84–89

Strassel S (2003) Simple metadata annotation specification version 5.0–may 14, 2003

Subramanya A, Petrov S, Pereira F (2010) Efficient graph-based semi-supervised learning of structured tagging models. In: Proceedings of the 2010 conference on empirical methods in natural language processing, EMNLP '10, USA. Association for Computational Linguistics, pp 167–176

Sun X, Morency LP, Okanohara D, Tsuruoka Y, Tsujii J (2008) Modeling latent-dynamic in shallow parsing: a latent conditional model with improved inference. In: Proceedings of the 22nd international conference on computational linguistics (Coling 2008), pp 841–848

Sun X, Sun S, Yin M, Yang H (2020) Hybrid neural conditional random fields for multi-view sequence labeling. Knowl-Based Syst 189:105151

Sutton C, McCallum A, Rohanimanesh K (2007) Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data. J Mach Learn Res 8(3)

Suzuki J, Isozaki H (2008) Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In: Proceedings of ACL-08: HLT, pp 665–673

Syed AZ, Aslam M, Martinez-Enriquez AM (2014) Associating targets with sentiunits: a step forward in sentiment analysis of urdu text. Artif Intell Rev 41(4):535–561

Symeonidis S, Effrosynidis D, Arampatzis A (2018) A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis. Expert Syst Appl 110:298–310

Taghipour K, Ng HT (2015) Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In: Proceedings of the 2015 conference of the North American chapter of the Association for Computational Linguistics: human language technologies, pp 314–323

Taylor P, Black AW, Caley R (1998) The architecture of the festival speech synthesis system. In: The third ESCA/COCOSDA workshop (ETRW) on speech synthesis

Telljohann H, Hinrichs E, Kübler S, Kübler R (2004) The tüba-d/z treebank: annotating german with a context-free backbone. In: Proceedings of the fourth international conference on language resources and evaluation (LREC 2004). Citeseer

Toutanova K, Cherry C (2009) A global model for joint lemmatization and part-of-speech prediction. In: Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP, pp 486–494

Toutanova K, Johnson M (2007) A Bayesian LDA-based model for semi-supervised part-of-speech tagging. Adv Neural Inf Process Syst 20:1521–1528

Toutanvoa K, Manning CD (2000) Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. 2000 Joint SIGDAT conference on empirical methods in natural language processing and very large corpora. China. Association for Computational Linguistics, Hong Kong, pp 63–70

Toutanova K, Klein D, Manning CD, Singer Y (2003) Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 conference of the North American chapter of the Association for Computational Linguistics on human language technology - vol 1, NAACL '03, USA. Association for Computational Linguistics, pp 173–180

Treviso MV, Shulby C, Aluísio SM (2016) Sentence segmentation in narrative transcripts from neuropsychological tests using recurrent convolutional neural networks. arXiv preprint arXiv:1610.00211

Treviso MV, Shulby CD, Aluisio SM (2017) Evaluating word embeddings for sentence boundary detection in speech transcripts

Tsochantaridis I, Hofmann T, Joachims T, Altun Y (2004) Support vector machine learning for interdependent and structured output spaces. In: Proceedings of the twenty-first international conference on Machine learning, pp 104

Tsuruoka Y, Tsujii J (2005) Bidirectional inference with the easiest-first strategy for tagging sequence data. In: Proceedings of the conference on human language technology and empirical methods in natural language processing, HLT '05, USA, pp 467–474. Association for Computational Linguistics

Ueberwasser S, Stark E (2017) What's up, Switzerland? A corpus-based research project in a multilingual country. Linguistik 84(5):105

Utgoff PE, Berkman NC, Clouse JA (1997) Decision tree induction based on efficient tree restructuring. Mach Learn 29(1):5–44. https://doi.org/10.1023/A:1007413323501

van Halteren H (2000) A default first order family weight determination procedure for WPDV models. In: Fourth conference on computational natural language learning and the second learning language in logic workshop

van Halteren H, Daelemans W, Zavrel J (2001) June) Improving accuracy in word class tagging through the combination of machine learning systems. Comput. Linguist. 27(2):199–229. https://doi.org/10.1162/089120101750300508

Van Halteren H (2000) Chunking with WPDV models. In: Fourth conference on computational natural language learning and the second learning language in logic workshop

Van Kerckvoorde CM (2019) An introduction to middle Dutch. De Gruyter Mouton

Vinyals O, Fortunato M, Jaitly N (2015) Pointer networks. In: Advances in neural information processing systems, pp 2692–2700

Wang P, Ng HT (2013) A beam-search decoder for normalization of social media text with application to machine translation. In: Proceedings of the 2013 conference of the North American chapter of the Association for Computational Linguistics: human language technologies, pp 471–481

Wang P, Qian Y, Soong FK, He L, Zhao H (2015) Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. CoRR abs/1510.06168. arXiv:1510.06168

Wei W, Wang Z, Mao X, Zhou G, Zhou P, Jiang S (2021) Position-aware self-attention based neural sequence labeling. Pattern Recogn 110:107636. https://doi.org/10.1016/j.patcog.2020.107636

Wilcox-O'Hearn A, Hirst G, Budanitsky A (2008) Real-word spelling correction with trigrams: a reconsideration of the mays, damerau, and mercer model. In: International conference on intelligent text processing and computational linguistics. Springer, pp 605–616

Winkler W (1990) String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In: Proceedings of the section on survey research methods

Wong F, Chao S (2010) isentenizer: an incremental sentence boundary classifier. In: Proceedings of the 6th international conference on natural language processing and knowledge engineering (NLPKE-2010). IEEE, pp 1–7

Wong DF, Chao LS, Zeng X (2014) isentenizer-: multilingual sentence boundary detection model. Sci World J 2014:1–10

Woolf BP (2009) Chapter 5 - communication knowledge. In: Woolf BP (ed) Building intelligent interactive tutors. Morgan Kaufmann, San Francisco, pp 136–182

Wu S, Cotterell R (2019) Exact hard monotonic attention for character-level transduction. arXiv preprint arXiv:1905.06319

Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, Zemel R, Bengio Y (2015a) Show, attend and tell: neural image caption generation with visual attention. In: International conference on machine learning. PMLR, pp 2048–2057

Xu K, Xia Y, Lee CH (2015b) Tweet normalization with syllables. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (vol 1: Long Papers), pp 920–928

Xue Z, Yin D, Davison BD (2011) Normalizing microtext. In: Workshops at the twenty-fifth AAAI conference on artificial intelligence. Citeseer

Yang Y, Eisenstein J (2013) A log-linear model for unsupervised text normalization. In: Proceedings of the 2013 conference on empirical methods in natural language processing, pp 61–72

Yang Z, Salakhutdinov R, Cohen W (2016) Multi-task cross-lingual sequence tagging from scratch. arXiv preprint arXiv:1603.06270

Yang Z, Salakhutdinov R, Cohen WW (2017) Transfer learning for sequence tagging with hierarchical recurrent networks. CoRR abs/1703.06345. arXiv:1703.06345

Yang J, Liang S, Zhang Y (2018) Design challenges and misconceptions in neural sequence labeling. arXiv preprint arXiv:1806.04470

Yang S, Wang Y, Chu X (2020) A survey of deep learning techniques for neural machine translation. arXiv preprint arXiv:2002.07526

Yildiz E, Tantuğ AC (2019) Morpheus: a neural network for jointly learning contextual lemmatization and morphological tagging. In: Proceedings of the 16th workshop on computational research in phonetics, phonology, and morphology, pp 25–34

Zalmout N, Habash N (2019) Joint diacritization, lemmatization, normalization, and fine-grained morphological tagging. arXiv preprint arXiv:1910.02267

Zalmout N, Habash N (2020) Utilizing subword entities in character-level sequence-to-sequence lemmatization models. In: Proceedings of the 28th international conference on computational linguistics, pp 4676–4682

Zeman D, Hajic J, Popel M, Potthast M, Straka M, Ginter F, Nivre J, Petrov S (2018) CoNLL 2018 shared task: multilingual parsing from raw text to universal dependencies. In: Proceedings of the CoNLL 2018 shared task: multilingual parsing from raw text to universal dependencies, pp 1–21

Zhai F, Potdar S, Xiang B, Zhou B (2017) Neural models for sequence chunking. arXiv preprint arXiv:1701.04027

Zhang T, Damerau F, Johnson DE (2001) Text chunking using regularized winnow. In: Proceedings of the 39th annual meeting of the Association for Computational Linguistics, pp 539–546

Zhang T, Damerau F, Johnson D (2002) Text chunking based on a generalization of winnow. J Mach Learn Res 2(Mar): 615–637

Zhang C, Baldwin T, Ho H, Kimelfeld B, Li Y (2013) Adaptive parser-centric text normalization. In: Proceedings of the 51st annual meeting of the Association for Computational Linguistics (vol 1: Long Papers), pp 1159–1168

Zhao J, Gao Q (2017) Annotation and detection of emotion in text-based dialogue systems with cnn. arXiv preprint arXiv:1710.00987

Zhao L, Qiu X, Zhang Q, Huang X (2019) Sequence labeling with deep gated dual path CNN. IEEE/ACM Trans Audio Speech Lang Process 27(12):2326–2335

Zhou G, Su J (2000) Error-driven HMM-based chunk tagger with context-dependent lexicon. In: 2000 Joint SIGDAT conference on empirical methods in natural language processing and very large corpora, pp 71–79

Zhou N, Wang X, Aw A (2017) Dynamic boundary detection for speech translation. In: 2017 Asia-pacific signal and information processing association annual summit and conference (APSIPA ASC), pp 651–656. IEEE

Zhou D, Zhang Z, Zhang ML, He Y (2018) Weakly supervised POS tagging without disambiguation. ACM Trans Asian Low-Resour Lang Inf Process (TALLIP) 17(4):1–19

Zhou H, Zhang Y, Li Z, Zhang M (2020) Is POS tagging necessary or even helpful for neural dependency parsing?